

# FINAL REPORT for NAG-1-2246

# Control Law — Control Allocation Interaction

F/A-18 PA Simulation Test-Bed

Dr. Wayne Durham & Mark Nelson

Department of Aerospace & Ocean Engineering

Virginia Polytechnic Institute & State University

Blacksburg, Virginia 24061

# 1 Introduction

This report documents the first stage of research into Control Law — Control Allocation Interactions. A three-year research effort was originally proposed:

1. Create a desktop flight simulation environment under which experiments related to the open questions may be conducted.
2. Conduct research to determine which aspects of control allocation have impact upon control law design that merits further research.
3. Conduct research into those aspects of control allocation identified above, and their impacts upon control law design.

Simulation code was written utilizing the F/A-18 airframe in the power-approach (PA) configuration. A dynamic inversion control law was implemented and used to drive a state-of-the-art control allocation subroutine.

# 2 Simulation

The airframe used was derived from the F/A-18 model already implemented in CASTLE. The airframe is not realistic, but is intended to be a test-bed for further research. The greatest area in which the test-bed simulation differs from the original airframe is in the treatment of control deflections. There are essentially two sets of control effectors:

1. The original control effectors in the F/A-18 airframe model. These are used only for initial trim and subsequent scheduling.

2. A duplicate set of control effectors that have linear effectiveness. This control set is the input to the control allocator, and the forces and moments they generate are superimposed on those of the bare airframe and original control set. Rate limits of the duplicate set are the no-load rate limits of the original controls. Position limits of the duplicate control effectors are referenced from the trim or scheduled positions of their counterparts in the original controls.

The rationale for incorporating a duplicate control set was to provide a constant, linear control effective matrix with flexibility for future variations and modifications. The control deflections are calculated for the trimmed/scheduled flight condition using the original F/A-18 nonlinear table lookups. The control deflections calculated from the allocator to produce desired moments use the control effectiveness matrix obtained from linearizing the F/A-18 aero database.

## **2.1 Simulation Code**

There are six files that are used in the simulation of the airframe: Aero.f, Aeropa.f, Control.f, Constants.f, Engine.f, and Alloc.f.

### **2.1.1 Aero.f**

The aero code first calls Aeropa.f to calculate the aerodynamics of the scheduled/trimmed flight condition. The code then combines the aerodynamics from the non-linear scheduled/trimmed flight condition and from the control deflections calculated in the allocator to produce the desired moments.

### 2.1.2 Aeropa.f

This code is taken from the F/A-18 simulation and modified slightly to include the control positions that are used in the table lookups later in the code. This is the only code that gives the airframe F/A-18 like characteristics. All added code is at the top of the RUN section.

### 2.1.3 Control.f

Stick and rudder pedal commands are taken as inputs and converted into an alpha command  $\alpha_{cmd}$ , beta command  $\beta_{cmd}$ , and a roll rate command  $p_{cmd}$ . These commands are input to a simple dynamic inversion control law that generates desired moments for the control allocation subroutine. First,  $\alpha_{cmd}$  and  $\beta_{cmd}$  are converted to desired accelerations  $\dot{w}_{des}$  and  $\dot{v}_{des}$ :

$$w_{cmd} = u \tan \alpha_{cmd}$$

$$\dot{w}_{des} = \lambda_w (w - w_{cmd})$$

$$v_{cmd} = V \sin \beta_{cmd}$$

$$\dot{v}_{des} = \lambda_v (v - v_{cmd})$$

Next,  $\dot{w}_{des}$  and  $\dot{v}_{des}$  are applied to inversions of the body-axis force equations (treating  $q$  and  $r$  as controls):

$$q_{cmd} = \frac{\dot{w}_{des} + pw - g \cos \theta \cos \phi - Z/m}{u}$$
$$r_{cmd} = \frac{-\dot{v}_{des} + pw + g \cos \theta \sin \phi + Y/m}{u}$$

These two inversions are made as perfect as possible by using actual aircraft states, and the last calculated values of the body-axis forces  $Y$  and  $Z$  from the

aerodynamic calculations. First-order responses are specified for the desired angular accelerations,

$$\dot{p}_{des} = \lambda_p (p - p_{cmd})$$

$$\dot{q}_{des} = \lambda_q (q - q_{cmd})$$

$$\dot{r}_{des} = \lambda_r (r - r_{cmd})$$

Finally, the desired body-axis moments, required to obtain the desired accelerations, are calculated from inversions of the body-axis moment equations:

$$\begin{aligned} C_{\ell_{des}}^c &= -C_{\ell}^a + \frac{I_{xz}\dot{p}_{des} - I_{xz}\dot{r}_{des} + (I_{xz} - I_{yy})qr - I_{xz}pq}{\dot{q}Sb} \\ C_{m_{des}}^c &= -C_m^a + \frac{I_{yy}\dot{q}_{des} + (I_{xx} - I_{zz})pr + I_{xz}(p^2 - r^2)}{\dot{q}S\bar{c}} \\ C_{n_{des}}^c &= -C_n^a + \frac{-I_{xz}\dot{p}_{des} + I_{xz}\dot{r}_{des} + (I_{yy} - I_{xx})pq + I_{xz}qr}{\dot{q}Sb} \end{aligned}$$

The moment coefficients  $C_{\ell}^a$ ,  $C_m^a$ , and  $C_n^a$  are the last calculated values of the body-axis moments. Since control-generated moments are superimposed on these values, they are the moments generated by the bare-airframe plus scheduled control deflections. The trimmed flight control deflections are used to calculate moments for the current flight condition to be used in the restoring algorithm. The attained moments are calculated next using the control deflections from the last time step for comparison purposes with the desired moments. The desired moments, along with the required inputs, are input to the allocator to produce the required control deflections. The last step is to check the control deflections against the limits and reset them accordingly.

#### 2.1.4 Constants.f

This section of code sets the model specific constants.

### 2.1.5 Engine.f

The engine model is taken from the Stevens & Lewis F-16 model [1].

### 2.1.6 Alloc.f

This code is the control allocator that produces required control deflections for desired moments. This code is explained in detail later.

## 3 Desktop Simulation

The F/A-18 PA model was first implemented on the UNIX-based CASTLE. The conversion of the simulation to the desktop PC required the CASTLE offline help menu provided with the PC version of CASTLE. Some additional steps were taken to complete the compilation of the airframe. The steps are as follows:

1. The directory structure from UNIX was copied to the CASTLE airframes folder.
2. A project was created in Microsoft Studio 6.0 following the F/A-18 project already included with the PC version of CASTLE. All custom builds were set up in the same way the F/A-18 project had them set up<sup>1</sup>. The custom builds were implemented on **symbols.sdf** and all the FTP data files.

3. In **symbols.sdf** the realtime CDF section was changed to resemble the

---

<sup>1</sup>The offline CASTLE help explains a different way of setting up custom builds, but did not work.

F/A-18 realtime CDF section in the corresponding **symbols.sdf**. The reason is just a difference in structure between the PC CASTLE and UNIX CASTLE.

## 4 Control Allocation Algorithm

### 4.1 Introduction

The control allocation algorithm is a FORTRAN implementation of the bisecting, edge-searching algorithm. The theory behind the allocation code is explained in detail in [2]. Following is a step-by-step explanation of the code. Line numbers correspond to those in the attached file “Alloc.f”.

### 4.2 Subroutine DA3

#### 4.2.1 Diagnostics

The sections of code that depend on the DIAGS flag are debugging tools that can be used to dump several relevant variables. Because a great volume of output is generated, the DIAGS flag should be used sparingly.

#### 4.2.2 Code Description

**Lines 0126–0146:** Array CSPHI is a table of sines and cosines of angles, beginning at  $45^\circ$  and proceeding through 20 bisections.

**Lines 0191–0208:** The desired moments are checked for zero length, and a vector of zero control deflections returned if they are; otherwise the

vector is normalized.

**Lines 0210–0219:** The initial rotation is performed using the transformation generated by subroutine DCGEN to align the desired moment direction  $y_{3_d}$  with the  $y_1$  axis. Subroutine **DCGEN** is an implementation of the method described in [2, Section 5.1]. Lines 0212–0219 perform the matrix multiplication,  $B_3 = T B_{3_{orig}}$ .

**Lines 0221–0231:** The controls that generate the moment with the maximum  $y_1$  component are found by examining the sign of the first row of B and setting the control to its maximum or minimum, depending on that sign. The controls are first set to  $\pm 1$  (object notation) and then set to their actual limits by subroutine **SETU**.

**Lines 0233–0243:** This section of code was added to deal with the finite precision of computer math. The variable TOL is a distance in moment space that is related to the smallest bisection angle to be used, at the distance from the origin of the vertex just determined (maximum  $y_1$  component). TOL is used in subsequent code to resolve near-zero numbers.

**Lines 0264–0265:** Subroutine **R20** solves the 2-D problem for the projection onto the current  $y_1$ - $y_2$  plane. **R20** returns the object-notation vector of controls of the intersecting edge in variable U1, the control that defines that edge in variable IU, and a  $\pm 1$  value in variable INFRONT that is +1 if the edge is in front of, and  $-1$  if it is behind the  $y_1$ - $y_2$  plane. The three variables TEMP2, TEMP3, and TEMP4 contain respectively

the sorted list of controls (ITHETA) with an additional zero between the two controls at the ends of the intersecting edge, the number of vertices in the list (NANGS), and the index of the position in the list of the additional zero (INDX). Finally, the logical variable ISVERTEX signals that the desired moment points directly at a vertex.

**Lines 0274–0293:** This section of code has no counterpart in reference [2]. It was added during debugging and found to improve the success rate of the algorithm (decrease the number of estimations required). The most recently found edges that were in front (Last In Front, LIF) and behind (Last In Back, LIB) are saved. Theoretically the last two edges found will be LIF and LIB, but in some cases they were not.

**Lines 0295–0299:** If **R20** reports a vertex in ISVERTEX, the controls that determine that vertex, and the saturation of the desired moment, are calculated by a call to **DOVERT**, and the subroutine is exited.

**Lines 0304–0322:** This section of code initializes several variables, including the rotation matrix **T22**.

**Lines 0333–0510:** This is the main loop, in which the 2-D problem is repeatedly solved for different rotations about the  $y_1$  axis.

**Lines 0335–0342:** Used during debugging, retained for possible future use.

**Lines 0344–0349:** Rotation about  $y_1$ . **B1** is the operative  $B$  matrix throughout. Code performs operation  $T \cdot B$ .

**Lines 0360–0364:** The last returned values of ITHETA, NANGS, and INDX are assigned to those variables to be saved when TEMP2, TEMP3, and TEMP4 are overwritten by **R20**.

**Lines 0366–0367:** Call to **R20** to solve the 2-D problem for the current orientation of B1 about the  $y_1$ -axis.

**Lines 0376–0395:** The edge identified by **R20** is assigned to LIF or LIB according to the sign of the variable INFRONT.

**Lines 0397–0401:** Another vertex check.

**Lines 0411–0495:** Executed when the most recent and the prior edges differ in sign of their  $y_3$  component, as indicated by the variables INFRONT and WASINF. This section of code is the implementation of the description given in [2, Section 5.3]. Through line 0436 the code is doing housekeeping and (possibly) diagnostics.

**Lines 0438–0457:** This section reflects a subtlety in the implementation of the algorithm not described in [2]. The prior edge was identified using a different  $B$  matrix than the most recent edge. All relevant information regarding the prior edge is contained in the saved variables ITHETA, NANGS, and INDX. At lines 0456–0457 a call is made to subroutine **GETEDGE**, which is also called as the last step of subroutine **R20**.

**Lines 0459–0478:** More last-in-front, last-in-back checking, and lines 0480–0484 deal with vertex checking.

**Lines 0486–0493:** Check the last two edges identified to see if they comprise the solution facet. If they do not, the LIF and LIB edges are checked. Both sets of edges are checked using sub-routine **ISFACET**, described below. Output from **ISFACET** consists of the logical ISOK, numbers of the two defining controls in IUOUT and JUOUT, and controls (in object notation) at three vertices of the facet as columns of the array U123.

**Lines 0518–0519:** If the variable ISOK is false, the correct facet has not been determined and the maximum number of bisections has been performed. One last check of LIF and LIB is performed.

**Lines 0520–0574:** If ISOK is true, the solution is calculated. Otherwise (lines 0572–0573) the solution is estimated.

**Lines 0521–0565:** A straightforward implementation of [2, Equations (13) and (14)]. M123 is the matrix  $[\hat{\mathbf{e}}_{3,1}(\mathbf{v}_1^y - \mathbf{v}_2^y)(\mathbf{v}_1^y - \mathbf{v}_3^y)]$  in [2, Equation (13)]; variables AA, BB, and CC correspond to  $\alpha_3$ ,  $C_{1,2}$ , and  $C_{1,3}$  respectively; and MTEMP is  $\mathbf{v}_1^y$ . The variable UDA is the same as  $\mathbf{u}^u$  in [2, Equation (14)], except that it has been scaled as necessary.

**Lines 0572–0573:** The estimator is called.

### 4.3 Subroutine DCGEN

This subroutine is a straightforward implementation of the initial transformation algorithm described in [2, Section 5.1].

**Lines 0789–0798:** The desired moments are normalized using double precision math.

**Lines 0810–0824:** If one or more of the leading components of the normalized moment vector are zero, the size of the problem is reduced.

**Lines 0829–0833:** The first row of the transformation matrix is set to the normalized desired moments.

**Lines 0837–0850:** The remaining terms are calculated in the three nested do-loops in [2, Equation (4)].

**Lines 0858–0868:** The last section of **DCGEN** ensures that the determinant of the transformation matrix is +1.

### 4.4 Subroutine R20

To find the edge that the desired moments direction is pointing to, the subroutine **R20** is implemented. The theory behind this subroutine is in [2, Section 5.2.2]. All calculations in this subroutine are done in the  $y_1$ - $y_2$  plane.

**Lines 0928–0961:** The  $y_2$  component of the point with the maximum  $y_1$  component (UMAX in object notation, XUMAX in control notation) is calculated to determine its sign. The desired moment is checked to see

if its direction points towards a vertex of the attainable moment subset.

If it is a vertex the subroutine is exited and the allocation carries on.

**Lines 0976–0992:** Implementation of [2, Step 1, Page 20]. The array THETA is the needed part of the set  $\mathcal{L}_\phi$ , and ITHETA that of  $\mathcal{L}_u$ . Once the angle is found,  $\pi$  is added or subtracted from it if the absolute value is greater than  $\pi/2$  and depending on the sign of the angle. In this way, the angles of just the vertices with positive  $y_1$  components are generated.

**Lines 0998–1010:** The angles are sorted in a clockwise or counter-clockwise manner starting with the vertex that has the largest  $y_2$  component. The manner in which they are sorted depends on the sign of the  $y_2$  component of the maximum vertex, recorded in SY.

**Lines 1013–1025:** A zero is inserted in THETA and ITHETA to mark the point at which the angle changes sign.

**Lines 1034–1036:** THETA, ITHETA, and NANGS (the number of angles generated) are sent to subroutine **GETEDGE** to finish the solution to the 2-D problem. Subroutine **GETEDGE** is provided separately so that it could be called independently from **DA3**, as described above.

## 4.5 Subroutine GETEDGE

This subroutine is part of the explanation in [2, Section 5.2.2].

**Lines 1090–1127:** The first loop in this subroutine is looking for a sign change in the  $y_2$  component between ordered vertices. Since the vertices

were sorted in the manner described, the solution edge will be the first one encountered in traversing the edges starting with the first vertex. The list is stepped through in the proper direction by the index  $IX = IX - SY$ . The previous  $y_2$  value is stored before the next  $y_2$  value is calculated. This new value is compared to the previous one determining whether the edge crosses the  $y_1$  axis. If the do loop continues, U2 is set to the next vertex by changing the sign of the control that is defining the current edge. The index is updated accordingly with the sign of  $y_2$  and the process starts again until the edge is found. The do loop is exited when a new point is found that has a different sign than the point before.

**Lines 1129–1181:** This section deals with possible failure of the previous loop to find an edge, as indicated by (SY.EQ.SSY). The starting values of relevant variables are restored, and the vertex list is traversed in the opposite direction. The first loop should always find the proper edge when **GETEDGE** is called from **R20**, but the first loop may fail when called from within **DA3**. The list is traversed in the opposite direction by the index  $IX = IX + SY$ . Implementation of this section of code was the reason for inserting a zero in the ordered list of vertices.

**Lines 1190–1200:** One or the other of the previous two loops will have identified U2 (a vertex in object notation) and JU (the number of the control that defines the edge). U2 is converted to control notation using the subroutine **SETU**. The third row of the B matrix is applied to the two

vertices that define the solution edge to determine the  $y_3$  component in moment space at the point where the  $y_2$  component of the edge is zero. If the  $y_3$  component is positive, the edge is described as “in front”, whereas if the  $y_3$  component is negative, the edge is “behind” the line defined by the direction of the desired moments  $\ell_3$ .

**Lines 1202–1217:** A final vertex check is made and the subroutine is exited.

#### 4.6 Subroutine DOVERT

**Lines 693–728:** If it was determined that the desired moments points directly to a vertex the subroutine **DOVERT** is called. **DOVERT** uses the maximum or minimum controls that make up the vertex and calculates the total moment from there, scaling it appropriately. The allocator subroutine is then exited and the simulation carries on. This case is rare during simulation, but may occur.

#### 4.7 Subroutine EST

The theory behind the estimator subroutine is explained in [2, Section 5.4.2].

**Lines 0604–629** The subroutine starts with the last two edges that the allocator had found and creates a facet by setting the appropriate control to  $-1$  or  $+1$ . SETU is used to assign actual control limits to these points which are then put into moment space using the control effectiveness matrix.

**Lines 631–669** An interpolation is then made with the estimated facet vertices to determine the solution.

**Lines 671–686** The moments are calculated using the estimated control positions and then scaled with the saturation limits.

## 4.8 Subroutine ISFACET

The subroutine is used to test the facet found by **DA3**. The subroutine uses the two defining controls from **DA3** to find a facet from scratch that these two controls define. This algorithm is the subject of reference [4].

**Lines 1236–1251:** Zeros are set in the appropriate positions of the vertex arrays so that two edges are defined for the facet. The dimension of the union (see [2, Section 4.2]) of the two edges is determined. If the union is not two dimensional, then the edges can not form a facet; **ISOK** is set to false and the subroutine is exited.

**Lines 1253–1320:** For the two dimensional case the routine begins to calculate from scratch the facet that is determined by the two defining controls. The method used is completely independent of the edge-searching method and is explained in [4].

**Lines 1255–1287:** This section of code was lifted from earlier FORTRAN implementations of the facet-searching allocation method described in reference [4]. The facet defined by the two controls is in the variable **TESTFACET**.

**Lines 1291–1311:** The facet **TESTFACET** is compared with the object **OBJ** that was generated by **R20**. If they are different, the

facet opposite TESTFACET (also generated by the same two controls) is tested (lines 1300–1311).

**Lines 1322–1336:** If the facet just found is the same facet as the one that was found from the allocator, then U123, which is the matrix whose columns correspond to controls that generate three of the vertices that make up the solution facet, is assembled and returned.

## 4.9 Miscellaneous Subroutines

### 4.9.1 MINNORM

The purpose of the minimum-norm restoring solution is to keep the controls as close to their trimmed control position as possible. The usual minimum-norm solution keeps the controls as close to zero as possible, however, in this application the zero position is redefined as the trimmed/scheduled control positions.

**Lines 1496–1531:** The subroutine is started by finding the total control position for the current time step and calculating the total attained moment.

**Lines 1533–1554:** If the control limits are zero, the routine is returned and no restoring takes place. Otherwise, the difference between the pseudo-inverse solution redefined at the trim condition, and the controls given by the allocation routine are used to find a delta control position that will drive the controls towards the trimmed position. This delta control position is scaled according to the control limits and a new restored control position is returned.

For more information on control restoring, refer to Bolling. [3, Ch. 4]

#### 4.9.2 SORTC

**Lines 1351–1492:** A sorting subroutine downloaded from the National Institute of Standards and Technology (NIST) GAMS (Guide to Available Mathematical Software) at <http://gams.nist.gov/>. This particular algorithm was chosen for its efficiency, and for the fact that it returns a sorted index vector along with the sorted vector.

#### 4.9.3 INVMAT3

**Lines 0740–767:** A brute force matrix inversion subroutine. Good only for  $3 \times 3$  matrices.

## 5 Verification Data

Sample runs are included to verify the airframe. The four tests cases used are a trimmed flight condition, a step in the longitudinal stick, and step doublets in the lateral stick and rudder pedals. The **MANGEN** command in CASTLE was implemented to produce the desired stick commands. Complete MATLAB files of the four cases are attached as `trim_dec11.mat`, `long_dec11.mat`, `lat_dec11.mat`, and `dir_dec11.mat`.

The plots include selected states of the airframe along with the trimmed/scheduled control positions and the allocated control positions.

Figure 1 shows the time histories of the six global controls in a trimmed flight condition at 8.1 degrees angle of attack, 1200 ft, and  $231.52 \frac{ft}{sec}$ . These settings

are the default when the airframe is loaded. Some settling of the controls to achieve steady state is noted.

Figure 2 shows time histories for a step input in longitudinal stick of 2.5 inches aft from center. The airframe was initialized to the trim conditions described above and the stick step implemented at  $time = 1sec$  for 1 second.

Figure 3 shows the time histories for a step doublet in lateral stick. The lateral stick was driven right 2 inches from center at  $time = 1sec$  for 1 second and then left of center 2 inches for 1 second.

Figure 4 shows the time histories for a step doublet in rudder pedals. The pedals were driven right 2 inches from center at  $time = 1sec$  for 1 second and then left of center 2 inches for 1 second.

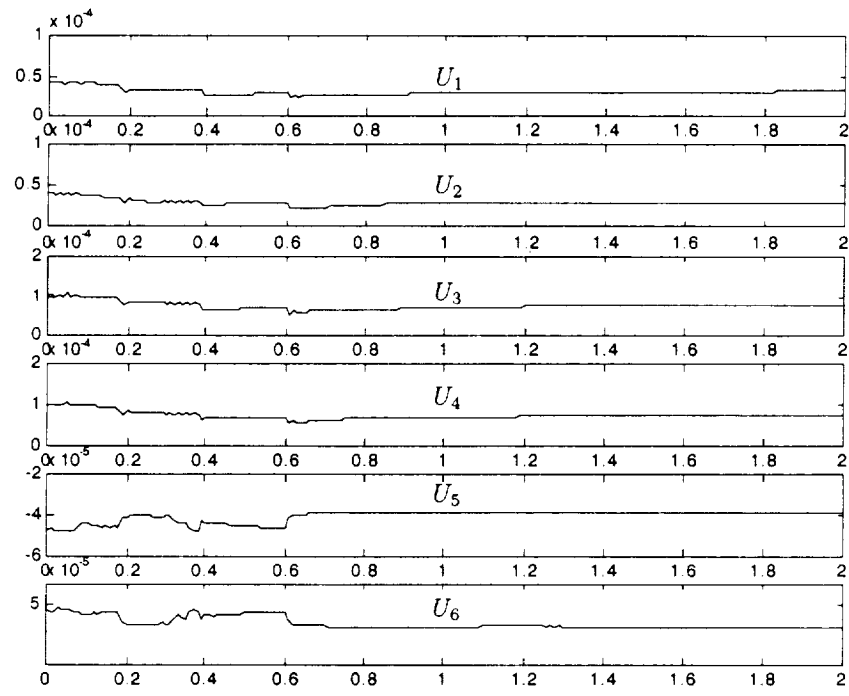


Figure 1: Global Control Deflections, Trimmed Flight (Degrees)

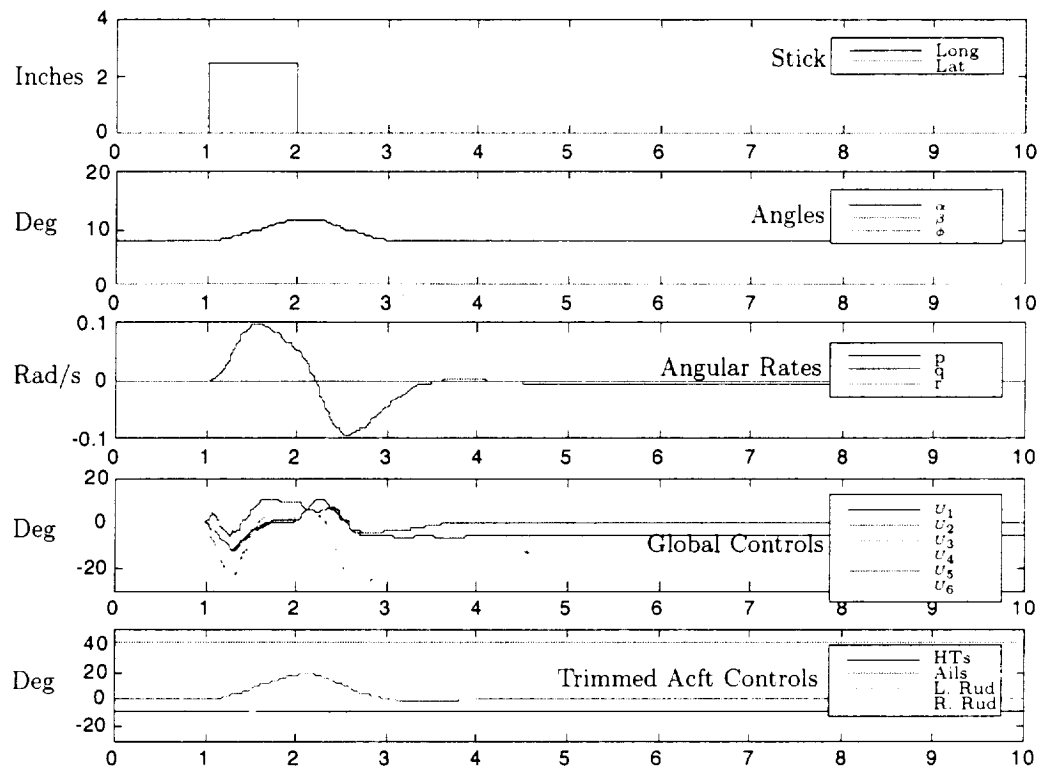


Figure 2: Longitudinal Stick Step Input

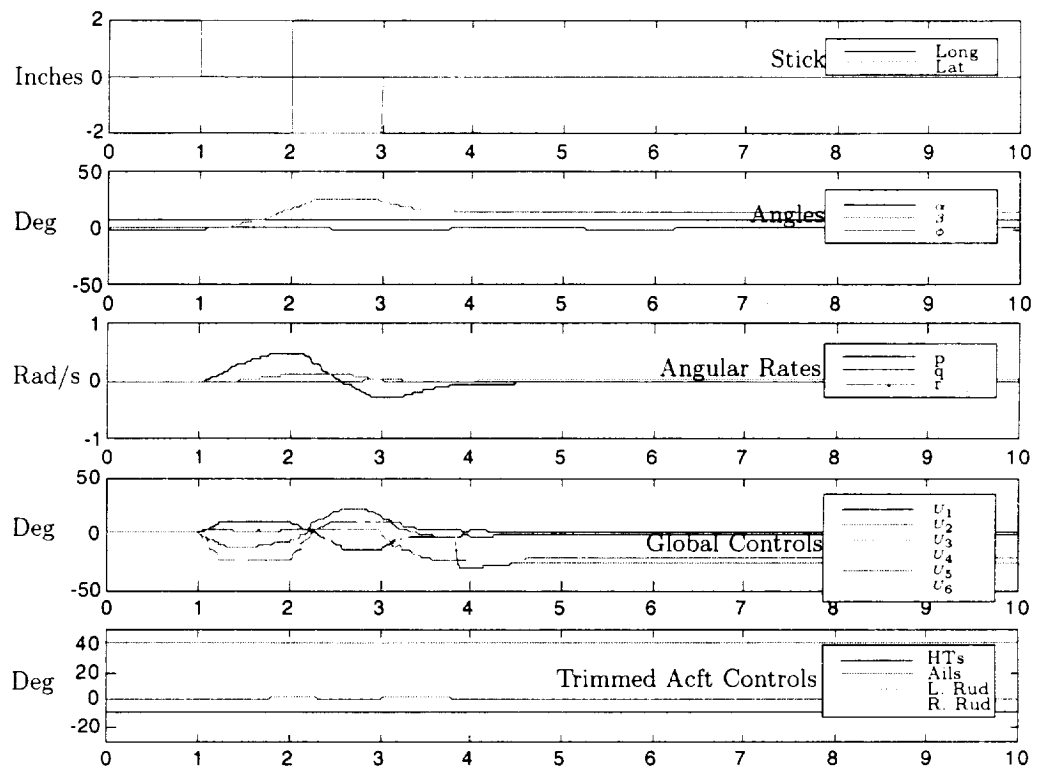


Figure 3: Lateral Stick Step Doublet

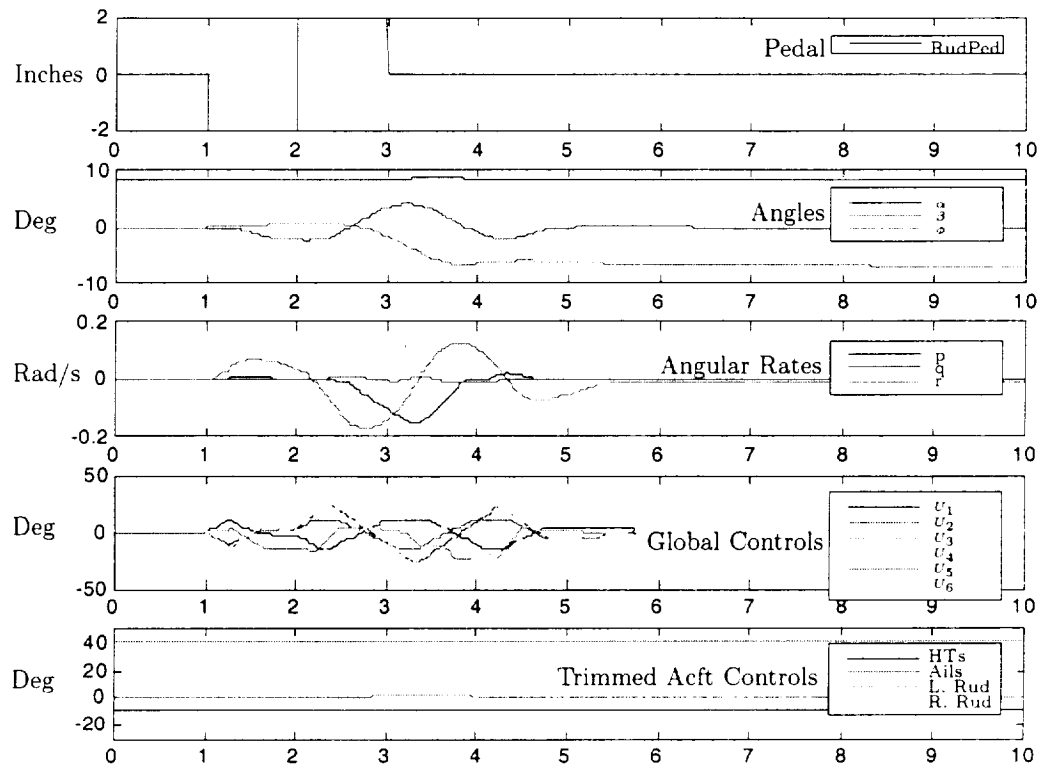


Figure 4: Lateral Rudder Pedal Step Doublet

## References

- [1] Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*, 1st ed: John Wiley & Sons, 1992, pp. 617.
- [2] Durham, W. C., "Computationally Efficient Control Allocation," *Guidance, Control, and Dynamics*, To appear, 2000 (copy attached).
- [3] Bolling, J.G., "Implementation of Constrained Control Allocation Techniques Using an Aerodynamic Model of an F-15 Aircraft" Master's Thesis, Dept. of Aerospace and Ocean Engineering, Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, May 1997.
- [4] Durham, W.C., "Attainable Moments for the Constrained Control Allocation Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No.6, 1994, pp.1371-1373.
- [5] Durham, W.C., "Constrained Control Allocation: Three Moment Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No.2, 1994, pp.330-336.
- [6] Scalera, K.R., "A Comparison of Control Allocation Methods for the F-15 ACTIVE Research Aircraft Utilizing Real-Time Piloted Simulations," MS Thesis, Virginia Polytechnic Institute & State University, 1999.

```
0001 C*****
0002 C
0003 C  TITLE:      DA3
0004 C
0005 C-----
0006 C
0007 C  FUNCTION:    3 Moment Control Allocator
0008 C              Direct Allocation for the 3 objective problem
0009 C              using bisecting edge searching alogorithm
0010 C
0011 C
0012 C-----
0013 C
0014 C  DESIGNED BY:   Bull Durham
0015 C
0016 C  CODED BY:     Kevin Scalera
0017 C
0018 C  MAINTAINED BY: VPI SIMULATIONS
0019 C
0020 C-----
0021 C
0022 C  MODIFICATION HISTORY:
0023 C
0024 C  DATE          PURPOSE          BY
0025 C  ====          =====          ==
0026 C
0027 C
0028 C-----
0029 C
0030 C              GLOSSARY
0031 C              =====
0032 C
0033 C  ASSIGNMENTS:
0034 C
0035 C  NONE
0036 C
0037 C-----
0038 C
0039 C  INPUTS:
0040 C
0041 C  IMODE          Sim. mode: -2=init,-1=reset,0=hold,1=ru  -----
0042 C
0043 C-----
0044 C
0045 C  OUTPUTS:
0046 C
0047 C  NONE
0048 C
0049 C-----
0050 C
0051 C  LOCALS:
0052 C
0053 C  NONE
0054 C
0055 C-----
0056 C
0057 C  OTHER LOCALS:
```

```
0058 C
0059 C NONE
0060 C-----
0061
0062      SUBROUTINE  DA3(UDA, SAT, IERR,
0063 C AS FUNCTIONS OF
0064      & B, MDES, U_MIN, U_MAX, M, NBI, TIME, DIAGS)
0065
0066 C-----
0067 C
0068 C  DECLARATION SECTION
0069 C
0070 C-----
0071      IMPLICIT NONE
0072 C      ** Parameters
0073
0074
0075 C      ** INPUTS:
0076
0077      INTEGER*4 IMODE
0078
0079 C      ** OTHER LOCALS:
0080
0081      BYTE CONPAR, CTLBUF
0082      LOGICAL*4 DIAGS, DIDSWITCH, INITIALIZED, ISOK, ISVERTEX, STUCK
0083      INTEGER*4 I, ICOUNT, IERR, INDX, ITHETA(21), IU, IUOUT
0084      INTEGER*4 IUTEMP(20), I_LIB, I_LIF, J, JU, JUOUT, K, M
0085      INTEGER*4 MAXSTEPS, NANGS, NBI, NMAX, STEPS, SY, TEMP2(21)
0086      INTEGER*4 TEMP3, TEMP4, U1(20), U123(20,3), U2(20), UMAX(20)
0087      INTEGER*4 U_LIB(20), U_LIF(20)
0088      REAL*4 MINV(3,3), MTEMP(3), MXMAX(3), COSPHI, CSPHI(2,20)
0089      REAL*4 ABC(3), NORM, PI, SAT, SINPHI, DET, AA, INFRONT, T(3,3)
0090      REAL*4 T22(2,2), BB, BTEMP(2), CC, TIME, TOL, TOLANG, B(3,20)
0091      REAL*4 M123(3,3), MAXNORM, UDA(20), B1(3,20), MD(3), MDES(3)
0092      REAL*4 U_MAX(20), U_MIN(20), WASINF, XU123(20,3), XUMAX(20)
0093      REAL*4 XUTEMP(20), Y
0094
0095 C-----
0096 C
0097 C  COMMON SECTION
0098 C
0099 C-----
0100
0101      COMMON/ SHELL1 / CONPAR( 424)
0102
0103 C-----
0104 C
0105 C  EQUIVALENCE SECTION
0106 C
0107 C-----
0108
0109 C      ** INPUTS:
0110
0111      EQUIVALENCE( CONPAR(1), IMODE)
0112
0113
0114
```

```

0115 C-----
0116 C
0117 C  DATA SECTION
0118 C
0119 C-----
0120
0121      DATA INITIALIZED/.FALSE./
0122      DATA PI/3.141592653589793/
0123 C
0124 C Table of cosines and sines of bisection angle
0125 C
0126      DATA CSPHI/
0127      & 7.071067811865475e-01,      7.071067811865476e-01,
0128      & 9.238795325112867e-01,      3.826834323650898e-01,
0129      & 9.807852804032304e-01,      1.950903220161282e-01,
0130      & 9.951847266721969e-01,      9.801714032956060e-02,
0131      & 9.987954562051724e-01,      4.906767432741801e-02,
0132      & 9.996988186962042e-01,      2.454122852291229e-02,
0133      & 9.999247018391445e-01,      1.227153828571993e-02,
0134      & 9.999811752826011e-01,      6.135884649154475e-03,
0135      & 9.999952938095762e-01,      3.067956762965976e-03,
0136      & 9.999988234517019e-01,      1.533980186284766e-03,
0137      & 9.999997058628822e-01,      7.669903187427045e-04,
0138      & 9.999999264657179e-01,      3.834951875713956e-04,
0139      & 9.999999816164293e-01,      1.917475973107033e-04,
0140      & 9.999999954041073e-01,      9.587379909597734e-05,
0141      & 9.999999988510269e-01,      4.793689960306688e-05,
0142      & 9.999999997127567e-01,      2.396844980841822e-05,
0143      & 9.999999999281892e-01,      1.198422490506971e-05,
0144      & 9.999999999820472e-01,      5.992112452642428e-06,
0145      & 9.999999999955118e-01,      2.996056226334661e-06,
0146      & 9.999999999988780e-01,      1.498028113169011e-06/
0147
0148 C-----
0149 C
0150 C  INITIALIZATION SECTION
0151 C
0152 C-----
0153
0154      IF( (IMODE.LE.-2) .OR. .NOT.INITIALIZED ) THEN
0155      ENDIF
0156 C-----
0157 C
0158 C  RESET SECTION
0159 C
0160 C-----
0161
0162      IF ((IMODE.LE.-1).OR.(.NOT.Initialized)) THEN
0163      Initialized = .TRUE.
0164 C
0165 C IERR = 0 FACET FOUND, ABC OK
0166 C IERR = 1 FACET NOT FOUND, INTERPOLATED SOLUTION
0167 C
0168      ENDIF
0169 C-----
0170 C
0171 C  RUN SECTION

```

```

0172 C
0173 C-----
0174 C
0175     IF (DIAGS) THEN
0176         WRITE(*, '(A50)') ' Entering  DAB'
0177         WRITE(*, '(A50)') ' Calling arguments'
0178         WRITE(*, '(A30,6E18.6)') ' *DAB* B(1,:) = ', (B(1,I), I=1,M)
0179         WRITE(*, '(A30,6E18.6)') ' *DAB* B(2,:) = ', (B(2,I), I=1,M)
0180         WRITE(*, '(A30,6E18.6)') ' *DAB* B(3,:) = ', (B(3,I), I=1,M)
0181         WRITE(*, '(A30,3E18.6)') ' *DAB* MDES = ', (MDES(I), I=1,3)
0182         WRITE(*, '(A30,6E14.6)') ' *DAB* U_MIN = ', (U_MIN(I), I=1,M)
0183         WRITE(*, '(A30,6E14.6)') ' *DAB* U_MAX = ', (U_MAX(I), I=1,M)
0184         WRITE(*, '(A30,13)') ' *DAB* N = ', M
0185         WRITE(*, '(A30,13)') ' *DAB* NBI = ', NBI
0186         WRITE(*, '(A30,6E14.6)') ' *DAB* TIME = ', TIME
0187     ENDIF
0188
0189     INFRONT = 1.0
0190
0191     NORM = 0.0
0192     DO I = 1,3
0193         NORM = NORM + MDES(I)*MDES(I)
0194     ENDDO
0195 C
0196     IF (NORM .EQ. 0.0) THEN
0197         IERR = 0
0198         SAT = 0.0
0199         DO I = 1,M
0200             UDA(I) = 0.0
0201         ENDDO
0202         RETURN
0203     ENDIF
0204
0205     NORM = SQRT(NORM)
0206     DO I = 1,3
0207         MD(I) = MDES(I)/NORM
0208     ENDDO
0209 C
0210     CALL DCGEN(T, MD)
0211 C
0212     DO I = 1,3
0213         DO J = 1,M
0214             B1(I,J) = 0.0
0215             DO K = 1,3
0216                 B1(I,J) = B1(I,J) + T(I,K)*B(K,J)
0217             ENDDO
0218         ENDDO
0219     ENDDO
0220 C
0221     DO I = 1,M
0222         IF (B1(1,I).EQ.0.) THEN
0223             UMAX(I) = 0
0224         ELSEIF (B1(1,I).LT.0.0) THEN
0225             UMAX(I) = -1
0226         ELSE
0227             UMAX(I) = 1
0228         ENDIF

```

```

0229      ENDDO
0230 C
0231      CALL SETU(XUMAX,UMAX,U_MIN,U_MAX,M)
0232 C
0233      TOLANG = CSPHI(2,MIN(20,2*NBI))
0234      DO I=1,3
0235          MXMAX(I)=0.
0236          DO J=1,M
0237              MXMAX(I) = MXMAX(I)+B1(I,J)*XUMAX(J)
0238          ENDDO
0239      ENDDO
0240      MAXNORM = SQRT(MXMAX(1)*MXMAX(1)
0241      &          +MXMAX(2)*MXMAX(2)
0242      &          +MXMAX(3)*MXMAX(3))
0243      TOL = MAXNORM*TOLANG
0244 C
0245      IF (DIAGS) THEN
0246          WRITE(*, '(//A50)') ' Preliminary Calcs'
0247          WRITE(*, '(A30,E18.6)') ' *DA3* NORM      = ', NORM
0248          WRITE(*, '(A30,3F14.6)') ' *DA3* MD        = ', (MD(I), I=1,3)
0249          WRITE(*, '(A30,3F14.6)') ' *DA3* T(1,:)     = ', (T(1,I), I=1,3)
0250          WRITE(*, '(A30,3F14.6)') ' *DA3* T(2,:)     = ', (T(2,I), I=1,3)
0251          WRITE(*, '(A30,3F14.6)') ' *DA3* T(3,:)     = ', (T(3,I), I=1,3)
0252          WRITE(*, '(A30,6E18.6)') ' *DA3* B1(1,:)    = ', (B1(1,I), I=1,M)
0253          WRITE(*, '(A30,6E18.6)') ' *DA3* B1(2,:)    = ', (B1(2,I), I=1,M)
0254          WRITE(*, '(A30,6E18.6)') ' *DA3* B1(3,:)    = ', (B1(3,I), I=1,M)
0255          WRITE(*, '(A30,6I3)') ' *DA3* UMAX        = ', (UMAX(I), I=1,M)
0256          WRITE(*, '(A30,6F14.6)') ' *DA3* XUMAX      = ', (XUMAX(I), I=1,M)
0257          WRITE(*, '(A30,E18.6)') ' *DA3* TOLANG     = ', TOLANG
0258          WRITE(*, '(A30,3E18.6)') ' *DA3* MXMAX      = ', (MXMAX(I), I=1,3)
0259          WRITE(*, '(A30,E18.6)') ' *DA3* MAXNORM     = ', MAXNORM
0260          WRITE(*, '(A30,E18.6)') ' *DA3* TOL         = ', TOL
0261          WRITE(*, '(//A50)') ' First call to R20'
0262      ENDIF
0263 C
0264      CALL R20(U1,IU,INFRONT,TEMP2,TEMP3,TEMP4,ISVERTEX,
0265      & B1,UMAX,XUMAX,U_MIN,U_MAX,TOL,M,DIAGS)
0266 C
0267      IF (DIAGS) THEN
0268          WRITE(*, '(//A50)') ' After 1st R20'
0269          WRITE(*, '(A30,7I3)') ' *DA3* TEMP2 (ITHETA) = ', (TEMP2(I), I=1,TEMP3+1)
0270          WRITE(*, '(A30,I3)') ' *DA3* TEMP3 (NANGS)  = ', TEMP3
0271          WRITE(*, '(A30,I3)') ' *DA3* TEMP4 (INDX)   = ', TEMP4
0272      ENDIF
0273 C
0274      IF (IU.NE.0) THEN
0275          IF (INFRONT.EQ.1.) THEN
0276              DO I=1,M
0277                  U_LIF(I) = U1(I)
0278              ENDDO
0279              I_LIF = IU
0280          ELSEIF (INFRONT.EQ.-1.) THEN
0281              DO I=1,M
0282                  U_LIB(I) = U1(I)
0283              ENDDO
0284              I_LIB = IU
0285          ENDIF

```

```

0286      IF (DIAGS) THEN
0287          WRITE(*, '(A50)') ' After 1st LIF/LIB'
0288          WRITE(*, '(A30,6I3)') ' *DA3* U_LIF = ', (U_LIF(I), I=1,M)
0289          WRITE(*, '(A30,I3)') ' *DA3* I_LIF = ', I_LIF
0290          WRITE(*, '(A30,6I3)') ' *DA3* U_LIB = ', (U_LIB(I), I=1,M)
0291          WRITE(*, '(A30,I3)') ' *DA3* I_LIB = ', I_LIB
0292      ENDIF
0293  ENDIF
0294 C
0295      IF (ISVERTEX) THEN
0296 C          WRITE(*,*) ' TIME = ', TIME, ' FIRST CALL TO R20'
0297          CALL DOVERT(UDA,SAT,U1,B,U_MIN,U_MAX,M,NORM)
0298          RETURN
0299      ENDIF
0300 C
0301 C
0302 C 1st rotation about x-axis
0303 C
0304      IF (M.GE.8) THEN
0305          ICOUNT = 1
0306      ELSE
0307          ICOUNT = 2
0308      ENDIF
0309          ICOUNT = 1
0310      COSPHI = CSPHI(1,ICOUNT)
0311      SINPHI = INFRONT*CSPHI(2,ICOUNT)
0312      T22(1,1) = COSPHI
0313      T22(1,2) = -SINPHI
0314      T22(2,1) = SINPHI
0315      T22(2,2) = COSPHI
0316      MAXSTEPS = 2*INT(ABS(PI/ASIN(SINPHI)))
0317      WASINF = INFRONT
0318      ISOK = .FALSE.
0319      NMAX = NBI + 1
0320      DIDSWITCH = .FALSE.
0321      STEPS = 0
0322      STUCK = .FALSE.
0323 C
0324      IF (DIAGS) THEN
0325          WRITE(*, '(A50)') ' Before Main Loop'
0326          WRITE(*, '(A30,E18.6)') ' *DA3* COSPHI = ', COSPHI
0327          WRITE(*, '(A30,E18.6)') ' *DA3* SINPHI = ', SINPHI
0328          WRITE(*, '(A30,I3)') ' *DA3* MAXSTEPS = ', MAXSTEPS
0329      ENDIF
0330 C
0331 C MAIN LOOP *****
0332 C
0333      DO WHILE ((ICOUNT.LT.NMAX).AND.(.NOT.ISOK))
0334 C
0335          STEPS = STEPS+1
0336          IF (STEPS.GE.MAXSTEPS) THEN
0337              STUCK = .TRUE.
0338 C              WRITE(*, '(A30)') ' ***** '
0339 C              WRITE(*, '(A30,I3)') ' *DA3* STUCK = ', STUCK
0340 C              WRITE(*, '(A30,E18.6)') ' *DA3* TIME = ', TIME
0341 C              WRITE(*, '(A30,/)') ' ***** '
0342          ENDIF

```

```

0343
0344      DO J = 1,M
0345          BTEMP(1) = T22(1,1)*B1(2,J) + T22(1,2)*B1(3,J)
0346          BTEMP(2) = T22(2,1)*B1(2,J) + T22(2,2)*B1(3,J)
0347          B1(2,J)   = BTEMP(1)
0348          B1(3,J)   = BTEMP(2)
0349      ENDDO
0350 C
0351      IF (DIAGS) THEN
0352          WRITE(*, '(/A50)') ' In DA3 DOWHILE'
0353          WRITE(*, '(A30,I3)') ' *DA3* ICOUNT = ', ICOUNT
0354          WRITE(*, '(A30,I3)') ' *DA3* STEPS = ', STEPS
0355          WRITE(*, '(A30,I3)') ' *DA3* MAXSTEPS = ', MAXSTEPS
0356          WRITE(*, '(A30,6E18.6)') ' *DA3* B1(2,:) = ', (B1(2,I), I=1,M)
0357          WRITE(*, '(A30,6E18.6)') ' *DA3* B1(3,:) = ', (B1(3,I), I=1,M)
0358      ENDIF
0359 C
0360      NANGS = TEMP3
0361      INDX = TEMP4
0362      DO I=1,21
0363          ITHETA(I) = TEMP2(I)
0364      ENDDO
0365 C
0366      CALL R20(U1,IU,INFRONT,TEMP2,TEMP3,TEMP4,ISVERTEX,
0367 & B1,UMAX,XUMAX,U_MIN,U_MAX,TOL,M,DIAGS)
0368 C
0369      IF (DIAGS) THEN
0370          WRITE(*, '(/A50)') ' After Loop R20'
0371          WRITE(*, '(A30,7I3)') ' *DA3* TEMP2 (ITHETA) = ', (TEMP2(I), I=1,TEMP3+1)
0372          WRITE(*, '(A30,I3)') ' *DA3* TEMP3 (NANGS) = ', TEMP3
0373          WRITE(*, '(A30,I3)') ' *DA3* TEMP4 (INDX) = ', TEMP4
0374      ENDIF
0375 C
0376      IF (IU.NE.0) THEN
0377          IF (INFRONT.EQ.1.) THEN
0378              DO I=1,M
0379                  U_LIF(I) = U1(I)
0380              ENDDO
0381              I_LIF = IU
0382          ELSEIF (INFRONT.EQ.-1.) THEN
0383              DO I=1,M
0384                  U_LIB(I) = U1(I)
0385              ENDDO
0386              I_LIB = IU
0387          ENDIF
0388          IF (DIAGS) THEN
0389              WRITE(*, '(/A50)') ' After Loop R20 LIF/LIB'
0390              WRITE(*, '(A30,6I3)') ' *DA3* U_LIF = ', (U_LIF(I), I=1,M)
0391              WRITE(*, '(A30,I3)') ' *DA3* I_LIF = ', I_LIF
0392              WRITE(*, '(A30,6I3)') ' *DA3* U_LIB = ', (U_LIB(I), I=1,M)
0393              WRITE(*, '(A30,I3)') ' *DA3* I_LIB = ', I_LIB
0394          ENDIF
0395      ENDIF
0396 C
0397      IF (ISVERTEX) THEN
0398 C          WRITE(*,*) ' TIME = ', TIME, ' LOOP CALL TO R20'
0399          CALL DOVERT(UDA,SAT,U1,B1,U_MIN,U_MAX,M,NORM)

```

```

0400      RETURN
0401      ENDIF
0402
0403      IF (DIAGS) THEN
0404          WRITE(*, '(//A50)') ' Before testing reversal'
0405          WRITE(*, '(A30,F14.6)') ' *DA3* INFRONT = ', INFRONT
0406          WRITE(*, '(A30,F14.6)') ' *DA3* WASINF = ', WASINF
0407      ENDIF
0408
0409      DIDSWITCH = .FALSE.
0410
0411      IF (INFRONT.NE.WASINF) THEN ! REVERSE DIRECTION
0412          IF (DIAGS) THEN
0413              WRITE(*, '(//A50)') ' Reversing'
0414              WRITE(*, '(A30,I3)') ' *DA3* Steps taken = ', STEPS
0415              WRITE(*, '(A30,F16.8)') ' *DA3* Angle = ', 180.*ASIN(SINPHI)/PI
0416              WRITE(*, '(A30,I3)') ' *DA3* MAXSTEPS = ', MAXSTEPS
0417              WRITE(*, '(A30,I3)') ' *DA3* STUCK = ', STUCK
0418          ENDIF
0419          DIDSWITCH = .TRUE.
0420          WASINF = INFRONT
0421          ICOUNT = ICOUNT+1
0422      C Bisection and next transformation
0423          COSPHI = CSPHI(1,ICOUNT)
0424          SINPHI = INFRONT*CSPHI(2,ICOUNT)
0425          T22(1,1) = COSPHI
0426          T22(1,2) = -SINPHI
0427          T22(2,1) = SINPHI
0428          T22(2,2) = COSPHI
0429          MAXSTEPS = 2*INT(ABS(PI/ASIN(SINPHI)))
0430          IF (DIAGS) THEN
0431              WRITE(*, '(//A50)') ' Bisection and next transformation'
0432              WRITE(*, '(A30,E18.6)') ' *DA3* COSPHI = ', COSPHI
0433              WRITE(*, '(A30,E18.6)') ' *DA3* SINPHI = ', SINPHI
0434              WRITE(*, '(A30,I5)') ' *DA3* MAXSTEPS = ', MAXSTEPS
0435          ENDIF
0436          STEPS = 0
0437      C Check last edge with new B1
0438          Y = 0.0
0439          DO I = 1,M
0440              Y = Y + B1(2,I)*XUMAX(I)
0441          ENDDO
0442          SY = 1
0443          IF (Y.LT.0.0) SY = -1
0444
0445          IF (DIAGS) THEN
0446              WRITE(*, '(//A50)') ' Before GETEDGE'
0447              WRITE(*, '(A30,6E18.6)') ' *DA3* B1(2,:) = ', (B1(2,I),I=1,M)
0448              WRITE(*, '(A30,6E18.6)') ' *DA3* B1(3,:) = ', (B1(3,I),I=1,M)
0449              WRITE(*, '(A30,6I3)') ' *DA3* UMAX = ', (UMAX(I),I=1,M)
0450              WRITE(*, '(A30,E18.6)') ' *DA3* Y = ', Y
0451              WRITE(*, '(A30,I3)') ' *DA3* SY = ', SY
0452              WRITE(*, '(A30,7I3)') ' *DA3* ITHETA = ', (ITHETA(I),I=1,NANGS+1)
0453              WRITE(*, '(A30,I3)') ' *DA3* NANGS = ', NANGS
0454          ENDIF
0455
0456          CALL GETEDGE(U2, JU, INFRONT, ISVERTEX,

```

```

0457      &      B1, UMAX, Y, SY, ITHETA, NANGS, U_MIN, U_MAX, INDX, TOL, M, DIAGS)
0458 C
0459      IF (JU.NE.0) THEN
0460          IF (INFRONT.EQ.1.) THEN
0461              DO I=1,M
0462                  U_LIF(I) = U2(I)
0463              ENDDO
0464              I_LIF = JU
0465          ELSEIF (INFRONT.EQ.-1.) THEN
0466              DO I=1,M
0467                  U_LIB(I) = U2(I)
0468              ENDDO
0469              I_LIB = JU
0470          ENDIF
0471          IF (DIAGS) THEN
0472              WRITE(*, '(A30)') ' After GETEDGE LIF/LIB'
0473              WRITE(*, '(A30,6I3)') ' *DA3* U_LIF = ', (U_LIF(I), I=1,M)
0474              WRITE(*, '(A30,I3)') ' *DA3* I_LIF = ', I_LIF
0475              WRITE(*, '(A30,6I3)') ' *DA3* U_LIB = ', (U_LIB(I), I=1,M)
0476              WRITE(*, '(A30,I3)') ' *DA3* I_LIB = ', I_LIB
0477          ENDIF
0478      ENDIF
0479 C
0480      IF (ISVERTEX) THEN
0481 C          WRITE(*,*) ' TIME = ', TIME, ' FOR GETEDGE'
0482          CALL DOVERT(UDA,SAT,U2,B,U_MIN,U_MAX,M,NORM)
0483          RETURN
0484      ENDIF
0485
0486      IF (JU.NE.0) THEN
0487          CALL ISFACET(ISOK, IUOUT, JUOUT, U123,
0488      &          IU, JU, U1, U2, B1, M, TOL)
0489          IF (.NOT.ISOK) CALL ISFACET(ISOK, IUOUT, JUOUT, U123,
0490      &          I_LIF, I_LIB, U_LIF, U_LIB, B1, M, TOL)
0491      ELSE
0492          ISOK = .FALSE.
0493      ENDIF
0494
0495      ENDIF ! IF (INFRONT.NE.WASINF) THEN
0496
0497 C
0498 C Must leave on a switch
0499 C
0500 C      IF ((ICOUNT.EQ.NMAX).AND.(.NOT.ISOK).AND.(.NOT.DIDSWITCH)) THEN
0501 C          NMAX = NMAX+1
0502 C      ENDIF
0503
0504      IF (STUCK) THEN
0505 C          WRITE(*, '(A50)') ' Stuck in DA3, exiting'
0506 C          WRITE(*, '(A30,F14.6)') ' TIME = ', TIME
0507          RETURN
0508      ENDIF
0509 C
0510      ENDDO ! End of do while statement
0511
0512 C END MAIN LOOP *****
0513 C

```

```

0514     IF (DIAGS) THEN
0515         WRITE(*, '(//A50)') ' Exited from DAB'
0516     ENDIF
0517
0518     IF (.NOT.ISOK) CALL ISFACET(ISOK, IUOUT, JUOUT, U123,
0519 & I_LIF, I_LIB, U_LIF, U_LIB, B1, M, TOL)
0520     IF (ISOK) THEN
0521         DO I=1,3
0522             DO J=1,M
0523                 IUTEMP(J)=U123(J,I)
0524             ENDDO
0525             CALL SETU(XUTEMP,IUTEMP,U_MIN,U_MAX,M)
0526             DO J=1,M
0527                 XU123(J,I)=XUTEMP(J)
0528             ENDDO
0529         ENDDO
0530 C
0531         DO I=1,3
0532             DO J=1,3
0533                 M123(I,J)=0.
0534             DO K=1,M
0535                 M123(I,J)=M123(I,J)+B(I,K)*XU123(K,J)
0536             ENDDO
0537         ENDDO
0538     ENDDO
0539 C
0540     DO I=1,3
0541         DO J=2,3
0542             M123(I,J)=M123(I,1)-M123(I,J)
0543         ENDDO
0544         MTEMP(I)=M123(I,1)
0545         M123(I,1)=MDES(I)
0546     ENDDO
0547 C
0548     CALL INVMAT3(M123,MINV,DET)
0549 C
0550     DO I=1,3
0551         ABC(I) = 0.
0552         DO J=1,3
0553             ABC(I) = ABC(I)+MINV(I,J)*MTEMP(J)
0554         ENDDO
0555     ENDDO
0556     AA = ABC(1)
0557     BB = ABC(2)
0558     CC = ABC(3)
0559     SAT = 1./AA
0560     IF (AA.LT.1.) AA = 1.
0561     DO I=1,M
0562         UDA(I) = (XU123(I,1)
0563 & +BB*(XU123(I,2)-XU123(I,1))
0564 & +CC*(XU123(I,3)-XU123(I,1)))/AA
0565     ENDDO
0566
0567     IERR = 0
0568 C
0569 C Call estimate subroutine to estimate solution if facet not found
0570 C

```

```
0571      ELSE
0572          CALL EST(UDA, SAT, IERR,
0573      & U_LIF, I_LIF, U_LIB, I_LIB, B1, U_MIN, U_MAX, NORM, M)
0574      ENDIF
0575
0576 C
0577      RETURN
0578      END
0579 C-----
0580
0581      SUBROUTINE EST(UDA, SAT, IERR,
0582      & U1, IU, U2, JU, B, U_MIN, U_MAX, NORM, M)
0583      IMPLICIT NONE
0584
0585 C INPUTS
0586      REAL*4 B(3,20), U_MAX(20), U_MIN(20), NORM
0587      INTEGER*4 U1(20), IU, U2(20), JU, M
0588
0589 C OUTPUTS
0590      REAL*4 SAT, UDA(20)
0591      INTEGER*4 IERR
0592
0593 C LOCALS
0594      REAL*4 XU1(20), XU2(20), XU3(20), XU4(20), XMOM(3)
0595      REAL*4 UPPER1(3), UPPER2(3), LOWER1(3), LOWER2(3), XNORM
0596      REAL*4 XK1, XK2, XK3, XV1(3), XV2(3), XW1(20), XW2(20)
0597      INTEGER*4 U3(20), U4(20)
0598
0599 C OTHER LOCALS
0600      INTEGER*4 I, J, K
0601
0602      IERR = 1
0603
0604      U1(IU) = -1
0605      U2(JU) = -1
0606      DO I=1,M
0607          U3(I) = U1(I)
0608          U4(I) = U2(I)
0609      ENDDO
0610      U3(IU) = 1
0611      U4(JU) = 1
0612
0613      CALL SETU(XU1,U1,U_MIN,U_MAX,M)
0614      CALL SETU(XU2,U2,U_MIN,U_MAX,M)
0615      CALL SETU(XU3,U3,U_MIN,U_MAX,M)
0616      CALL SETU(XU4,U4,U_MIN,U_MAX,M)
0617
0618      DO I=1,3
0619          LOWER1(I) = 0.
0620          LOWER2(I) = 0.
0621          UPPER1(I) = 0.
0622          UPPER2(I) = 0.
0623          DO J=1,M
0624              LOWER1(I) = LOWER1(I)+B(I,J)*XU1(J)
0625              LOWER2(I) = LOWER2(I)+B(I,J)*XU2(J)
0626              UPPER1(I) = UPPER1(I)+B(I,J)*XU3(J)
0627              UPPER2(I) = UPPER2(I)+B(I,J)*XU4(J)
```

```
0628      ENDDO
0629      ENDDO
0630
0631      IF (LOWER1(2).NE.UPPER1(2)) THEN
0632          XK1 = LOWER1(2)/(LOWER1(2)-UPPER1(2))
0633      ELSE
0634          XK1 = 0.
0635      ENDIF
0636      IF (LOWER2(2).NE.UPPER2(2)) THEN
0637          XK2 = LOWER2(2)/(LOWER2(2)-UPPER2(2))
0638      ELSE
0639          XK2 = 0.
0640      ENDIF
0641
0642
0643      DO I=1,3
0644          XV1(I) = XK1*UPPER1(I)+(1.-XK1)*LOWER1(I)
0645          XV2(I) = XK2*UPPER2(I)+(1.-XK2)*LOWER2(I)
0646      ENDDO
0647
0648      IF (XV2(3).NE.XV1(3)) THEN
0649          XK3 = XV2(3)/(XV2(3)-XV1(3))
0650      ELSE
0651          XK3 = 0.
0652      ENDIF
0653
0654      DO I=1,M
0655          XW1(I) = XK1*XU3(I)+(1.-XK1)*XU1(I)
0656          XW2(I) = XK2*XU4(I)+(1.-XK2)*XU2(I)
0657      ENDDO
0658
0659      DO I=1,M
0660          UDA(I) = XK3*XW1(I)+(1.-XK3)*XW2(I)
0661      ENDDO
0662
0663
0664      DO I=1,3
0665          XMOM(I) = 0.
0666          DO J=1,M
0667              XMOM(I) = XMOM(I)+B(I,J)*UDA(J)
0668          ENDDO
0669      ENDDO
0670
0671      XNORM = SQRT(XMOM(1)*XMOM(1)
0672      &          +XMOM(2)*XMOM(2)
0673      &          +XMOM(3)*XMOM(3))
0674
0675      IF (XNORM.NE.0.) THEN
0676          SAT = NORM/XNORM
0677          XNORM = SAT
0678      ELSE
0679          SAT = 0.
0680      ENDIF
0681
0682      IF (XNORM.GT.1.) XNORM = 1.
0683
0684      DO I = 1,M
```

```

0685      UDA(I) = XNORM*UDA(I)
0686      ENDDO
0687
0688      RETURN
0689      END
0690
0691 C-----
0692
0693      SUBROUTINE DOVERT(UDA,SAT,
0694 & U1,B,U_MIN,U_MAX,M,NORM)
0695      IMPLICIT NONE
0696
0697      REAL*4 UDA(20), SAT, U_MIN(20), U_MAX(20), B(3,20), NORM
0698      INTEGER*4 U1(20), M
0699
0700      REAL*4 XMOM(3), XNORM
0701      INTEGER*4 I, J
0702
0703 C      WRITE(*,*) ' VERTEX'
0704
0705      CALL SETU(UDA,U1,U_MIN,U_MAX,M)
0706
0707      DO I=1,3
0708          XMOM(I) = 0.
0709          DO J=1,M
0710              XMOM(I) = XMOM(I)+B(I,J)*UDA(J)
0711          ENDDO
0712      ENDDO
0713
0714
0715      XNORM = SQRT(XMOM(1)*XMOM(1)
0716 &              +XMOM(2)*XMOM(2)
0717 &              +XMOM(3)*XMOM(3))
0718
0719      SAT = NORM/XNORM
0720      XNORM = SAT
0721      IF (XNORM.GT.1.) XNORM = 1.
0722
0723      DO I = 1,M
0724          UDA(I) = XNORM*UDA(I)
0725      ENDDO
0726
0727      RETURN
0728      END
0729
0730 C-----
0731
0732      SUBROUTINE INVMAT3(MATIN,MATOUT,DET)
0733
0734      IMPLICIT NONE
0735      INTEGER*4 I, J
0736      REAL*4 DET, MATIN(3,3), MATOUT(3,3)
0737 C
0738 C Zero out the output matrix
0739 C
0740      DO I = 1,3
0741          DO J = 1,3

```

```

0742          MATOUT(I,J)=0.0
0743          ENDDO
0744          ENDDO
0745 C
0746 C Calculate the determinant of the input matrix
0747 C
0748          DET = MATIN(1,1)*MATIN(2,2)*MATIN(3,3)
0749          &    + MATIN(1,2)*MATIN(2,3)*MATIN(3,1)
0750          &    + MATIN(1,3)*MATIN(2,1)*MATIN(3,2)
0751          &    - MATIN(1,3)*MATIN(2,2)*MATIN(3,1)
0752          &    - MATIN(1,2)*MATIN(2,1)*MATIN(3,3)
0753          &    - MATIN(1,1)*MATIN(2,3)*MATIN(3,2)
0754 C
0755 C Find the matrix inverse
0756 C
0757          IF (DET.NE.0.0) THEN
0758              MATOUT(1,1) = (MATIN(2,2)*MATIN(3,3)-MATIN(2,3)*MATIN(3,2))/DET
0759              MATOUT(1,2) = -(MATIN(1,2)*MATIN(3,3)-MATIN(1,3)*MATIN(3,2))/DET
0760              MATOUT(1,3) = (MATIN(1,2)*MATIN(2,3)-MATIN(1,3)*MATIN(2,2))/DET
0761              MATOUT(2,1) = -(MATIN(2,1)*MATIN(3,3)-MATIN(2,3)*MATIN(3,1))/DET
0762              MATOUT(2,2) = (MATIN(1,1)*MATIN(3,3)-MATIN(1,3)*MATIN(3,1))/DET
0763              MATOUT(2,3) = -(MATIN(1,1)*MATIN(2,3)-MATIN(1,3)*MATIN(2,1))/DET
0764              MATOUT(3,1) = (MATIN(2,1)*MATIN(3,2)-MATIN(2,2)*MATIN(3,1))/DET
0765              MATOUT(3,2) = -(MATIN(1,1)*MATIN(3,2)-MATIN(1,2)*MATIN(3,1))/DET
0766              MATOUT(3,3) = (MATIN(1,1)*MATIN(2,2)-MATIN(1,2)*MATIN(2,1))/DET
0767          ENDIF
0768
0769
0770
0771          RETURN
0772          END
0773
0774 C-----
0775
0776          SUBROUTINE DCGEN(T, MD)
0777
0778          IMPLICIT NONE
0779          REAL*4 MD(3)
0780          REAL*4 T(3,3)
0781          INTEGER*4 MLOCAL
0782          REAL*8 V(3), VLEN(3), VNORM, XDOT_DC
0783          REAL*4 DETNUM, AMIN_DC
0784          INTEGER*4 J, JCOL, KCOL, I, MOM_FLAG, IROW
0785          REAL*8 DTOL
0786 C
0787 C Calculate the norm of the moments
0788 C
0789          DTOL = 1.D-5
0790          VNORM = DSQRT(dble(MD(1))*dble(MD(1))
0791          &            +dble(MD(2))*dble(MD(2))
0792          &            +dble(MD(3))*dble(MD(3)))
0793 C
0794 C Normalize the desired moments
0795 C
0796          DO I = 1,3
0797              V(I) = dble(MD(I))/VNORM
0798          ENDDO

```

```
0799 C
0800 C Zero out the transformation matrix
0801 C
0802     DO I = 1,3
0803         DO J = 1,3
0804             T(I,J) = 0.0
0805         ENDDO
0806     ENDDO
0807 C
0808 C Check to see if V(I),3 to 1 is approx equal to zero => reduce size of problem
0809 C
0810     DO I = 3,1,-1
0811         IF (ABS(V(I)) .LE. DTOL) THEN
0812             T(I,I) = 1.0
0813         ELSE
0814             GOTO 5
0815         ENDIF
0816     ENDDO
0817 C
0818 C T(1,1) = 1.0 or -1.0 for rotation about x-axis (depends on direction of rot.)
0819 C
0820 5     IF (I .EQ. 1) THEN
0821         T(I,I) = 1.0
0822         IF (dble(MD(1)) .LT. 0.0D0) T(I,I) = -1.0
0823         RETURN
0824     ENDIF
0825 C
0826 C Set the 1st row of T equal to the normalized desired moments and
0827 C calculate the square of each of these values
0828 C
0829     MLOCAL = I
0830     DO I = 1,3
0831         T(1,I) = V(I)
0832         VLEN(I) = V(I)*V(I)
0833     ENDDO
0834 C
0835 C Developing orthogonal tranformation with V as 1st row
0836 C
0837     DO JCOL = 1,MLOCAL-1
0838         IROW = MLOCAL + 1 - JCOL
0839         T(IROW,JCOL) = sngl(DSQRT(1.0D0 - VLEN(JCOL)))
0840         DO KCOL = JCOL+1,MLOCAL
0841             XDOT_DC = T(1,JCOL)*T(1,KCOL)
0842             IF (IROW .NE. MLOCAL) THEN
0843                 DO I = IROW+1,MLOCAL
0844                     XDOT_DC = XDOT_DC + T(I,JCOL)*T(I,KCOL)
0845                 ENDDO
0846             ENDIF
0847             T(IROW,KCOL) = -XDOT_DC/T(IROW,JCOL)
0848             VLEN(KCOL) = VLEN(KCOL) + dble(T(IROW,KCOL))*dble(T(IROW,KCOL))
0849         ENDDO
0850     ENDDO
0851 C
0852 C Tricky stuff here!
0853 C
0854     DETNUM = int(mod(MLOCAL,4)/2.0)
0855 C
```

---

0856 C Necessary to do, but not easy to explain

0857 C

```

0858     IF (DETNUM.EQ.0.0) THEN
0859         IF (T(1,MLOCAL) .LT. 0.0) THEN
0860             T(2,MLOCAL-1) = -T(2,MLOCAL-1)
0861             T(2,MLOCAL)   = -T(2,MLOCAL)
0862         ENDIF
0863     ELSE
0864         IF (T(1,MLOCAL) .GT. 0.0) THEN
0865             T(2,MLOCAL-1) = -T(2,MLOCAL-1)
0866             T(2,MLOCAL)   = -T(2,MLOCAL)
0867         ENDIF
0868     ENDIF
0869
0870     RETURN
0871     END

```

0872

0873 C-----

0874

0875 SUBROUTINE SETU(XU\_SETU,IU\_SETU,U\_MIN,U\_MAX,M)

0876

0877 IMPLICIT NONE

0878 REAL\*4 U\_MAX(20), U\_MIN(20)

0879 INTEGER\*4 IMODE, M

0880 INTEGER\*4 I, IU\_SETU(20)

0881 REAL\*4 XU\_SETU(20)

0882

0883 DO I = 1,M

0884 IF (IU\_SETU(I) .EQ. 1) THEN

0885 XU\_SETU(I) = U\_MAX(I)

0886 ELSEIF (IU\_SETU(I) .EQ.-1) THEN

0887 XU\_SETU(I) = U\_MIN(I)

0888 ELSE

0889 XU\_SETU(I) = 0.

0890 ENDIF

0891 ENDDO

0892

0893 RETURN

0894 END

0895

0896 C-----

0897

```

0898     SUBROUTINE  R20(U1,IU,INFRONT,ITHETA,NANGS,INDX,ISVERTEX,
0899     & B1,UMAX,XUMAX,U_MIN,U_MAX,TOL,M,DIAGS)

```

0900

0901 IMPLICIT NONE

0902 REAL\*4 U\_MIN(20), U\_MAX(20)

0903 INTEGER\*4 IMODE, ITHETA(21), IU, UMAX(20), M, N, NANGS, SY

0904 INTEGER\*4 U1(20)

0905 REAL\*4 PIOVR2, B1(3,20), PI, XU(20), Y, XUMAX(20)

0906 REAL\*4 INFRONT, TOL

0907 REAL\*4 THETA(21)

0908 LOGICAL\*4 ISVERTEX, DIAGS

0909 INTEGER\*4 INDX, SSY

0910 INTEGER\*4 I, J

0911 REAL\*4 ANG, YPREV, XU1(20), K, Z, Z1, Z2

0912 C

```

0913      DATA PIOVR2/1.570796326794897/
0914      DATA PI/3.141592653589793/
0915 C
0916      IF (DIAGS) THEN
0917          WRITE(*, '(A50)') ' Entering R20'
0918          WRITE(*, '(A50)') ' Calling arguments'
0919          WRITE(*, '(A30,6E18.6)') ' *R20* B1(1,:) = ', (B1(1,I), I=1,M)
0920          WRITE(*, '(A30,6E18.6)') ' *R20* B1(2,:) = ', (B1(2,I), I=1,M)
0921          WRITE(*, '(A30,6E18.6)') ' *R20* B1(3,:) = ', (B1(3,I), I=1,M)
0922          WRITE(*, '(A30,6I3)') ' *R20* UMAX = ', (UMAX(I), I=1,M)
0923          WRITE(*, '(A30,6F14.6)') ' *R20* XUMAX = ', (XUMAX(I), I=1,M)
0924          WRITE(*, '(A30,E18.6)') ' *R20* TOL = ', TOL
0925          WRITE(*, '(A30,I3)') ' *R20* N = ', M
0926      ENDIF
0927 C
0928 C Calculate Y
0929 C
0930      Y = 0.0
0931      DO I = 1,M
0932          U1(I) = UMAX(I)
0933          Y = Y + B1(2,I)*XUMAX(I)
0934      ENDDO
0935 C VERTEX CHECK
0936
0937      ISVERTEX = .FALSE.
0938      IF (ABS(Y).LT.TOL) THEN
0939          Y = 0.
0940          Z = 0.
0941          DO I=1,M
0942              Z = Z+B1(3,I)*XUMAX(I)
0943          ENDDO
0944          IF (ABS(Z).LT.TOL) THEN
0945 C              WRITE(*,*) 'Z = ',Z,' TOL = ',TOL,' P20, ABS(Z).LT.TOL'
0946              ISVERTEX = .TRUE.
0947              IU = 0
0948 C              INFRONT = 0.
0949              NANGS = 0
0950              RETURN
0951          ENDIF
0952      ENDIF
0953 C END VERTEX CHECK
0954 C
0955      IF (ABS(Y).LT.TOL) THEN
0956          SY = 0
0957      ELSEIF (Y.LT.0.0) THEN
0958          SY = -1
0959      ELSE
0960          SY = 1
0961      ENDIF
0962 C
0963      IF (DIAGS) THEN
0964          WRITE(*, '(A50)') ' First calculations'
0965          WRITE(*, '(A30,E18.6)') ' *R20* Y = ', Y
0966          WRITE(*, '(A30,I3)') ' *R20* SY = ', SY
0967      ENDIF
0968 C
0969 C Get the angle

```

```

0970 C
0971 C
0972     IF (DIAGS) THEN
0973         WRITE(*, '(//A50)') ' Getting angles'
0974     ENDIF
0975 C
0976     NANGS = 0
0977     DO I=1, M
0978         ANG = ATAN2(-B1(1,I),B1(2,I))
0979         IF (ABS(ANG).GT.PIOVR2) THEN
0980             IF (ANG.LT.0.) THEN
0981                 ANG = ANG+PI
0982             ELSE
0983                 ANG = ANG-PI
0984             ENDIF
0985         ENDIF
0986         NANGS = NANGS+1
0987         THETA(NANGS) = ANG
0988         ITHETA(NANGS) = I
0989         IF (DIAGS) THEN
0990             WRITE(*, '(A30,I3,A10,E18.6)') ' *R20* I = ', I, ' ANG = ', ANG
0991         ENDIF
0992     ENDDO
0993 C
0994
0995 C THETA and ITHETA now sorted by control number
0996 C Sort THETA by magnitude. ITHETA gets shuffled the same way
0997 C
0998     IF (DIAGS) THEN
0999         WRITE(*, '(//A50)') ' Before sorting'
1000         WRITE(*, '(A30,I3)') ' *R20* NANGS = ', NANGS
1001         WRITE(*, '(A30,6I3)') ' *R20* ITHETA = ', (ITHETA(I), I=1,NANGS)
1002         WRITE(*, '(A30,6E18.6)') ' *R20* THETA = ', (THETA(I), I=1,NANGS)
1003     ENDIF
1004     CALL SORTC(THETA, ITHETA, NANGS, THETA, ITHETA)
1005     IF (DIAGS) THEN
1006         WRITE(*, '(//A50)') ' After sorting'
1007         WRITE(*, '(A30,I3)') ' *R20* NANGS = ', NANGS
1008         WRITE(*, '(A30,6I3)') ' *R20* ITHETA = ', (ITHETA(I), I=1,NANGS)
1009         WRITE(*, '(A30,6E18.6)') ' *R20* THETA = ', (THETA(I), I=1,NANGS)
1010     ENDIF
1011 C FIND INDEX OF ZERO
1012 C
1013     DO I=1,NANGS
1014         IF (THETA(I).GT.0.) THEN
1015             INDX = I
1016             DO J=NANGS,I,-1
1017                 THETA(J+1) = THETA(J)
1018                 ITHETA(J+1) = ITHETA(J)
1019             ENDDO
1020             THETA(I) = 0.
1021             ITHETA(I) = 0
1022             GOTO 193
1023         ENDIF
1024     ENDDO
1025 193 CONTINUE
1026 C

```

```

1027      IF (DIAGS) THEN
1028          WRITE(*, '(/A50)') ' After indexing'
1029          WRITE(*, '(A30,I3)')      ' *R20* INDX      = ', INDX
1030          WRITE(*, '(A30,I13)')      ' *R20* ITHETA   = ', (ITHETA(I), I=1,NANGS+1)
1031          WRITE(*, '(A30,7E18.6)') ' *R20* THETA     = ', (THETA(I), I=1,NANGS+1)
1032      ENDIF
1033 C
1034      CALL GETEDGE(U1, IU, INFRONT, ISVERTEX,
1035 C AS FUNCTIONS OF
1036      & B1, UMAX, Y, SY, ITHETA, NANGS, U_MIN, U_MAX, INDX, TOL, M, DIAGS)
1037 C
1038      IF (DIAGS) THEN
1039          WRITE(*, '(/A50)') ' Exiting R20'
1040          WRITE(*, '(A30,6I3)')      ' *R20* U1          = ', (U1(I), I=1,M)
1041          WRITE(*, '(A30,I3)')      ' *R20* IU          = ', IU
1042          WRITE(*, '(A30,5I14.6)') ' *R20* INFRONT     = ', INFRONT
1043          WRITE(*, '(A30,I3)')      ' *R20* ISVERTEX    = ', ISVERTEX
1044      ENDIF
1045 C
1046      RETURN
1047      END
1048
1049 C-----
1050
1051      SUBROUTINE  GETEDGE(U2, JU, INFRONT, ISVERTEX,
1052 C AS FUNCTIONS OF
1053      & B1, UMAX, Y, SY, ITHETA, NANGS, U_MIN, U_MAX, INDX, TOL, M, DIAGS)
1054
1055      IMPLICIT NONE
1056
1057      INTEGER*4 M, SY, UMAX(20), INDX, NANGS, ITHETA(21)
1058      REAL*4 B1(3,20), U_MAX(20), U_MIN(20), XU(20), Y
1059      REAL*4 INFRONT, TOL
1060      INTEGER*4 U2(20), JU, SSY
1061      LOGICAL*4 ISVERTEX, DIAGS, STUCK
1062      INTEGER*4 I, J, ICOUNT, IX
1063      REAL*4 XK, XU2(20), YPREV, Z, Z1, Z2, SAVE_Y, MOM(3)
1064 C
1065      IF (DIAGS) THEN
1066          WRITE(*, '(/A50)') ' Entering GETEDGE'
1067      ENDIF
1068 C
1069      DO I=1,M
1070          U2(I) = UMAX(I)
1071      ENDDO
1072 C
1073      SAVE_Y = Y
1074 C
1075      IX = INDX-SY
1076      SSY = SY
1077 C
1078      STUCK = .FALSE.
1079      ICOUNT = 0
1080 C
1081      IF (DIAGS) THEN
1082          WRITE(*, '(/A50)') ' Beginning GETEDGE DOWHILE'
1083          WRITE(*, '(A30,6I3)')      ' *GETEDGE* U2          = ', (U2(I), I=1,M)

```

```

1084      WRITE(*, '(A30,E18.6)') ' *GETEDGE* SAVE_Y = ', SAVE_Y
1085      WRITE(*, '(A30,I3)') ' *GETEDGE* IX = ', IX
1086      WRITE(*, '(A30,I3)') ' *GETEDGE* SSY = ', SSY
1087      WRITE(*, '(A30,I3)') ' *GETEDGE* NANGS = ', NANGS
1088  ENDIF
1089 C
1090  DO WHILE ((SY.EQ.SSY).AND.(IX.GT.0).AND.(IX.LE.NANGS))
1091      ICOUNT = ICOUNT+1
1092      YPREV = Y
1093      JU = ITHETA(IX)
1094      Y = Y-U2(JU)*B1(2,JU)*(U_MAX(JU)-U_MIN(JU))
1095      SSY = 1
1096      IF (Y.LT.0.0) SSY = -1
1097      IF (ABS(Y).LT.TOL) THEN
1098          Y = 0.
1099          SSY = 0
1100      ENDIF
1101      U2(JU) = -U2(JU)
1102      IF (SY.NE.0.) THEN
1103          IX = IX-SY
1104      ELSE
1105          IX = IX-1
1106      ENDIF
1107 C
1108      ICOUNT = ICOUNT+1
1109      IF (ICOUNT.GT.20) STUCK = .TRUE.
1110      IF (DIAGS) THEN
1111          WRITE(*, '(/A50)') ' Bottom of GETEDGE DOWHILE'
1112          WRITE(*, '(A30,I3)') ' *GETEDGE* ICOUNT = ', ICOUNT
1113          WRITE(*, '(A30,I3)') ' *GETEDGE* STUCK = ', STUCK
1114          WRITE(*, '(A30,E18.6)') ' *GETEDGE* YPREV = ', YPREV
1115          WRITE(*, '(A30,I3)') ' *GETEDGE* JU = ', JU
1116          WRITE(*, '(A30,E18.6)') ' *GETEDGE* Y = ', Y
1117          WRITE(*, '(A30,I3)') ' *GETEDGE* SSY = ', SSY
1118          WRITE(*, '(A30,I3)') ' *GETEDGE* SY.EQ.SSY = ', (SY.EQ.SSY)
1119          WRITE(*, '(A30,6I3)') ' *GETEDGE* U2 = ', (U2(I), I=1,M)
1120          WRITE(*, '(A30,I3)') ' *GETEDGE* IX = ', IX
1121      ENDIF
1122      IF (STUCK) THEN
1123          PAUSE('Stuck in GETEDGE in DOWHILE')
1124          STUCK = .FALSE.
1125      ENDIF
1126 C
1127  ENDDO
1128 C
1129  IF (SY.EQ.SSY) THEN
1130  C
1131      STUCK = .FALSE.
1132      ICOUNT = 0
1133  C
1134      IF (DIAGS) THEN
1135          WRITE(*, '(/A50)') ' Beginning Alt. GETEDGE DOWHILE'
1136          WRITE(*, '(A30,6I3)') ' *GETEDGE* U2 = ', (U2(I), I=1,M)
1137          WRITE(*, '(A30,E18.6)') ' *GETEDGE* SAVE_Y = ', SAVE_Y
1138          WRITE(*, '(A30,I3)') ' *GETEDGE* IX = ', IX
1139          WRITE(*, '(A30,I3)') ' *GETEDGE* SSY = ', SSY
1140          WRITE(*, '(A30,I3)') ' *GETEDGE* NANGS = ', NANGS

```

```

1141         ENDIF
1142 C
1143         Y = SAVE_Y
1144         IX = INDX+SY
1145         DO WHILE ((SY.EQ.SSY).AND.(IX.GT.0).AND.(IX.LE.NANGS+1))
1146             YPREV = Y
1147             JU = ITHETA(IX)
1148             Y = Y-U2(JU)*B1(2,JU)*(U_MAX(JU)-U_MIN(JU))
1149             SSY = 1
1150             IF (Y.LT.0.0) SSY = -1
1151             IF (ABS(Y).LT.TOL) THEN
1152                 Y = 0.
1153                 SSY = 0
1154             ENDIF
1155             U2(JU) = -U2(JU)
1156             IF (SY.NE.0.) THEN
1157                 IX = IX+SY
1158             ELSE
1159                 IX = IX+1
1160             ENDIF
1161 C
1162             ICOUNT = ICOUNT+1
1163             IF (ICOUNT.GT.20) STUCK = .TRUE.
1164             IF (DIAGS) THEN
1165                 WRITE(*, '(A50)') ' Bottom of Alt. GETEDGE DOWHILE'
1166                 WRITE(*, '(A30,I3)') ' *GETEDGE* ICOUNT = ', ICOUNT
1167                 WRITE(*, '(A30,I3)') ' *GETEDGE* STUCK = ', STUCK
1168                 WRITE(*, '(A30,E18.6)') ' *GETEDGE* YPREV = ', YPREV
1169                 WRITE(*, '(A30,I3)') ' *GETEDGE* JU = ', JU
1170                 WRITE(*, '(A30,E18.6)') ' *GETEDGE* Y = ', Y
1171                 WRITE(*, '(A30,I3)') ' *GETEDGE* SSY = ', SSY
1172                 WRITE(*, '(A30,I3)') ' *GETEDGE* SY.EQ.SSY = ', (SY.EQ.SSY)
1173                 WRITE(*, '(A30,6I3)') ' *GETEDGE* U2 = ', (U2(I), I=1,M)
1174                 WRITE(*, '(A30,I3)') ' *GETEDGE* IX = ', IX
1175             ENDIF
1176             IF (STUCK) THEN
1177                 PAUSE('Stuck in GETEDGE in Alt. DOWHILE')
1178             ENDIF
1179 C
1180         ENDDO
1181     ENDIF
1182 C
1183     IF (SY.EQ.SSY) THEN
1184         JU = 0
1185 C         INFRONT = 0.
1186         ISVERTEX = .FALSE.
1187         RETURN
1188     ENDIF
1189 C
1190     XK = YPREV/(YPREV-Y)
1191     CALL SETU(XU2, U2, U_MIN, U_MAX, M)
1192
1193     Z2 = 0.
1194     DO I=1,M
1195         Z2 = Z2+B1(3,I)*XU2(I)
1196     ENDDO
1197     Z1 = Z2-U2(JU)*B1(3,JU)*(U_MAX(JU)-U_MIN(JU))

```

```

1198      Z = XK*Z2+(1.-XK)*Z1
1199      INFRONT = 1.
1200      IF (Z.LT.0.) INFRONT = -1.
1201 C
1202      ISVERTEX = ((Y.EQ.0.) .AND. (ABS(Z).LT.TOL))
1203 C
1204      IF (DIAGS) THEN
1205          WRITE(*, '(/ASC)) ' Leaving GETEDGE'
1206          WRITE(*, '(A30,6I3)') ' *GETEDGE* U2 = ', (U2(I), I=1,M)
1207          WRITE(*, '(A30,6F14.6)') ' *GETEDGE* XU2 = ', (XU2(I), I=1,M)
1208          WRITE(*, '(A30,I3)') ' *GETEDGE* JU = ', JU
1209          WRITE(*, '(A30,E18.6)') ' *GETEDGE* XK = ', XK
1210          WRITE(*, '(A30,E18.6)') ' *GETEDGE* Z2 = ', Z2
1211          WRITE(*, '(A30,E18.6)') ' *GETEDGE* Z1 = ', Z1
1212          WRITE(*, '(A30,E18.6)') ' *GETEDGE* Z = ', Z
1213          WRITE(*, '(A30,F14.6)') ' *GETEDGE* INFRONT = ', INFRONT
1214          WRITE(*, '(A30,I3)') ' *GETEDGE* ISVERTEX = ', ISVERTEX
1215      ENDIF
1216 C
1217      RETURN
1218      END
1219
1220 C-----
1221
1222      SUBROUTINE ISFACET(ISOK, IUOUT, JUOUT, U123,
1223      &      IU, JU, U1, U2, B, M, TOL)
1224 C
1225      IMPLICIT NONE
1226
1227      LOGICAL*4 ISOK
1228      INTEGER*4 IU, JU, M, U1(20), U2(20), IUOUT, JUOUT, U123(20,3)
1229      REAL*4 B(3,20), TOL
1230 C LOCALS
1231      INTEGER*4 UX1(20), UX2(20), I, J, K, II, JJ, KK
1232      REAL*4 THEMAT(2,2), MATINV(2,2), MATDET, T2(2), T1(3), TESTFACET(20)
1233      REAL*4 TEMP
1234      INTEGER*4 DIM, OBJ(20), UDEF(20), ITF(20), THEOBJ(20)
1235
1236      ISOK = .FALSE.
1237      DO I=1,M
1238          UX1(I) = U1(I)
1239          UX2(I) = U2(I)
1240      ENDDO
1241      U1(IU) = 0
1242      U2(JU) = 0
1243      DIM = 0
1244      DO I = 1,M
1245          OBJ(I) = U1(I)
1246          IF ((I.EQ.IU) .OR. (I.EQ.JU) .OR. (U2(I).NE.U1(I))) THEN
1247              OBJ(I) = 0
1248              DIM = DIM + 1
1249              UDEF(DIM) = I
1250          ENDIF
1251      ENDDO
1252 C
1253      IF (DIM.EQ.2) THEN
1254          ISOK = .TRUE.

```

```

1255      DO II=1,3
1256          JJ = MOD(II,3)+1
1257          KK = MOD(JJ,3)+1
1258          THEMAT(1,1) = B(II,UDEF(1))
1259          THEMAT(1,2) = B(JJ,UDEF(1))
1260          THEMAT(2,1) = B(II,UDEF(2))
1261          THEMAT(2,2) = B(JJ,UDEF(2))
1262          MATDET = THEMAT(1,1)*THEMAT(2,2)-THEMAT(1,2)*THEMAT(2,1)
1263          IF (MATDET.NE.0.) THEN
1264              MATINV(1,1) = THEMAT(2,2)/MATDET
1265              MATINV(1,2) = -THEMAT(1,2)/MATDET
1266              MATINV(2,2) = THEMAT(1,1)/MATDET
1267              MATINV(2,1) = -THEMAT(2,1)/MATDET
1268              T2(1) = -MATINV(1,1)*B(KK,UDEF(1)) - MATINV(1,2)*B(KK,UDEF(2))
1269              T2(2) = -MATINV(2,1)*B(KK,UDEF(1)) - MATINV(2,2)*B(KK,UDEF(2))
1270              T1(KK) = 1.
1271              T1(II) = T2(1)
1272              T1(JJ) = T2(2)
1273          DO I=1,M
1274              TESTFACET(I) = 0.
1275              ITF(I) = 0
1276              UDEF(I) = 0
1277              DO J=1,3
1278                  TESTFACET(I) = TESTFACET(I)+T1(J)*B(J,I)
1279              ENDDO ! DO J=1,3
1280              IF (ABS(TESTFACET(I)).LT.TOL) THEN
1281                  ITF(I) = 0
1282              ELSEIF (TESTFACET(I).GT.0.) THEN
1283                  ITF(I) = 1
1284              ELSEIF (TESTFACET(I).LT.0.) THEN
1285                  ITF(I) = -1
1286              ENDIF
1287          ENDDO ! DO I=1,M
1288 C
1289          DIM = 0
1290 C
1291          DO I=1,M
1292              THEOBJ(I) = OBJ(I)
1293              IF ((OBJ(I).EQ.0) .OR. (ITF(I).EQ.0) .OR. (ITF(I).NE.OBJ(I))) THEN
1294                  THEOBJ(I) = 0
1295                  DIM = DIM + 1
1296                  UDEF(DIM) = I
1297              ENDIF
1298          ENDDO ! DO I=1,M
1299 C
1300          IF (DIM.NE.2) THEN
1301              DIM = 0
1302              J = 0
1303              DO I=1,M
1304                  THEOBJ(I) = OBJ(I)
1305                  IF ((OBJ(I).EQ.0) .OR. (ITF(I).EQ.0) .OR. (-ITF(I).NE.OBJ(I))) THEN
1306                      THEOBJ(I) = 0
1307                      DIM = DIM + 1
1308                      UDEF(DIM) = I
1309                  ENDIF
1310              ENDDO ! DO I=1,M
1311          ENDIF ! IF (DIM.NE.2) THEN

```

```

1312 C
1313             IF (DIM.NE.2) ISOK = .FALSE.
1314
1315             GOTO 194
1316
1317             ENDIF ! IF (MATDET.NE.0.) THEN
1318             ENDDO ! DO II=1,3
1319 194         CONTINUE
1320             ENDIF ! IF (DIM.EQ.2)
1321 C
1322             IF (ISOK) THEN
1323                 IUOUT = UDEF(1)
1324                 JUOUT = UDEF(2)
1325                 DO I=1,M
1326                     DO J=1,3
1327                         U123(I,J) = OBJ(I)
1328                     ENDDO
1329                 ENDDO
1330                 U123(IUOUT,1) = 1
1331                 U123(JUOUT,1) = 1
1332                 U123(IUOUT,2) = -1
1333                 U123(JUOUT,2) = 1
1334                 U123(IUOUT,3) = 1
1335                 U123(JUOUT,3) = -1
1336             ELSE
1337                 IUOUT = 0
1338                 JUOUT = 0
1339                 DO I=1,M
1340                     DO J=1,3
1341                         U123(I,J) = 0
1342                     ENDDO
1343                 ENDDO
1344             ENDIF
1345
1346             RETURN
1347             END
1348
1349 C-----
1350
1351             SUBROUTINE    SORTC(X,IY,N,XS,IYC)
1352
1353             IMPLICIT NONE
1354
1355             INTEGER*4 I, IL_SORTC(36), IMED, IP1, IPR, ITY, IU_SORTC(36)
1356             INTEGER*4 IY(20), IYC(20), J, JMI, JMK, K, L, LMI, M, MID, N
1357             INTEGER*4 NM1
1358             REAL*4 HOLD, AMED, TX, X(20), XS(20)
1359 C
1360 C         CHECK THE INPUT ARGUMENTS FOR ERRORS
1361 C
1362             IPR=11
1363             IF(N.LT.1)GOTO50
1364             IF(N.EQ.1)GOTO55
1365             HOLD=X(1)
1366             DO60I=2,N
1367             IF(X(I).NE.HOLD)GOTO90
1368 60         CONTINUE

```

```

1369 C      WRITE(*, 9)HOLD
1370      DO61I=1,N
1371      XS(I)=X(I)
1372      IYC(I)=IY(I)
1373      61 CONTINUE
1374      RETURN
1375 C      50 WRITE(*,15)
1376 C      WRITE(*,47)N
1377      50 RETURN
1378 C      55 WRITE(*,18)
1379      55 XS(1)=X(1)
1380      IYC(1)=IY(1)
1381      RETURN
1382      90 CONTINUE
1383      9 FORMAT(1X , '***** NON-FATAL DIAGNOSTIC--THE FIRST INPUT ARGUME
1384      INT (A VECTOR) TO THE SORTC SUBROUTINE HAS ALL ELEMENTS = ',E15.8,'
1385      1 *****')
1386      15 FORMAT(1X , '***** FATAL ERROR--THE SECOND INPUT ARGUMENT TO THE
1387      1 SORTC SUBROUTINE IS NON-POSITIVE *****')
1388      18 FORMAT(1X , '***** NON-FATAL DIAGNOSTIC--THE SECOND INPUT ARGUME
1389      INT TO THE SORTC SUBROUTINE HAS THE VALUE 1 *****')
1390      47 FORMAT(1X , '***** THE VALUE OF THE ARGUMENT IS ',I8 , ' *****')
1391 C
1392 C-----START POINT-----
1393 C
1394 C      COPY THE VECTOR X INTO THE VECTOR XS
1395      DO100I=1,N
1396      XS(I)=X(I)
1397      100 CONTINUE
1398 C
1399 C      COPY THE VECTOR IY INTO THE VECTOR IYC
1400 C
1401      DO150I=1,N
1402      IYC(I)=IY(I)
1403      150 CONTINUE
1404 C
1405 C      CHECK TO SEE IF THE INPUT VECTOR IS ALREADY SORTED
1406 C
1407      NM1=N-1
1408      DO200I=1,NM1
1409      IP1=I+1
1410      IF(XS(I) .LE. XS(IP1))GOTO200
1411      GOTO250
1412      200 CONTINUE
1413      RETURN
1414      250 M=1
1415      I=1
1416      J=N
1417      305 IF(I.GE.J)GOTO370
1418      310 K=I
1419      MID=(I+J)/2
1420      AMED=XS(MID)
1421      IMED=IYC(MID)
1422      IF(XS(I) .LE. AMED)GOTO320
1423      XS(MID)=XS(I)
1424      IYC(MID)=IYC(I)
1425      XS(I)=AMED

```

```
1426      IYC(I)=IMED
1427      AMED=XS(MID)
1428      IMED=IYC(MID)
1429  320  L=J
1430      IF(XS(J).GE.AMED)GOTO340
1431      XS(MID)=XS(J)
1432      IYC(MID)=IYC(J)
1433      XS(J)=AMED
1434      IYC(J)=IMED
1435      AMED=XS(MID)
1436      IMED=IYC(MID)
1437      IF(XS(I).LE.AMED)GOTO340
1438      XS(MID)=XS(I)
1439      IYC(MID)=IYC(I)
1440      XS(I)=AMED
1441      IYC(I)=IMED
1442      AMED=XS(MID)
1443      IMED=IYC(MID)
1444      GOTO340
1445  330  XS(L)=XS(K)
1446      IYC(L)=IYC(K)
1447      XS(K)=TX
1448      IYC(K)=ITY
1449  340  L=L-1
1450      IF(XS(L).GT.AMED)GOTO340
1451      TX=XS(L)
1452      ITY=IYC(L)
1453  350  K=K+1
1454      IF(XS(K).LT.AMED)GOTO350
1455      IF(K.LE.L)GOTO330
1456      LMI=L-I
1457      JMK=J-K
1458      IF(LMI.LE.JMK)GOTO360
1459      IL_SORTC(M)=I
1460      IU_SORTC(M)=L
1461      I=K
1462      M=M+1
1463      GOTO380
1464  360  IL_SORTC(M)=K
1465      IU_SORTC(M)=J
1466      J=L
1467      M=M+1
1468      GOTO380
1469  370  M=M-1
1470      IF(M.EQ.0)RETURN
1471      I=IL_SORTC(M)
1472      J=IU_SORTC(M)
1473  380  JMI=J-I
1474      IF(JMI.GE.11)GOTO310
1475      IF(I.EQ.1)GOTO305
1476      I=I-1
1477  390  I=I+1
1478      IF(I.EQ.J)GOTO370
1479      AMED=XS(I+1)
1480      IMED=IYC(I+1)
1481      IF(XS(I).LE.AMED)GOTO390
1482      K=I
```

```

1483   395 XS(K+1)=XS(K)
1484       IYC(K+1)=IYC(K)
1485       K=K-1
1486       IF(AMED.LT.XS(K))GOTO395
1487       XS(K+1)=AMED
1488       IYC(K+1)=IMED
1489       GOTO 390
1490
1491       RETURN
1492       END
1493
1494 C-----
1495
1496       SUBROUTINE   MINNORM(UMINNORM, SCALE,
1497 C AS A FUNCTION OF
1498       & P, U, UTRIM, MTRIM, UDA, B, U_MIN, U_MAX, M, TIME, XSCALE)
1499
1500       IMPLICIT NONE
1501 C       ** Parameters
1502
1503
1504 C       ** INPUTS:
1505
1506       REAL*4 P(20,3), U(20), UTRIM(20), UDA(20)
1507       REAL*4 B(3,20), MTRIM(3), U_MAX(20), U_MIN(20)
1508       REAL*4 TIME, XSCALE
1509       INTEGER*4 M, IMODE
1510
1511 C       ** OUTPUTS:
1512
1513       REAL*4 UMINNORM(20), SCALE
1514
1515 C       ** LOCALS:
1516
1517       REAL*4 UKDA(20), UP(20), UDELTA(20), M_ATT(3)
1518       INTEGER*4 I, J, K
1519 C
1520       SCALE = 1.
1521
1522       DO I=1,M
1523           UKDA(I) = U(I) + UDA(I)
1524       ENDDO
1525
1526       DO I=1,3
1527           M_ATT(I)=0.
1528           DO J=1,M
1529               M_ATT(I)=M_ATT(I)+B(I,J)*UKDA(J)
1530           ENDDO
1531       ENDDO
1532
1533       DO I=1,M
1534           IF ((U_MIN(I).EQ.0.).OR.(U_MAX(I).EQ.0.)) SCALE = 0.
1535           UKDA(I) = U(I) + UDA(I)
1536           UP(I) = UTRIM(I)
1537           DO J=1,3
1538               UP(I) = UP(I) + P(I,J)*(M_ATT(J)-MTRIM(J))
1539           ENDDO

```

```
1540      UDELTA(I) = SCALE*(UP(I)-UKDA(I))
1541      IF (UDELTA(I).NE.0.) THEN
1542          IF ((UDELTA(I).GT.U_MAX(I)).AND.(U_MAX(I).GT.0.)) THEN
1543              SCALE = U_MAX(I)/UDELTA(I)
1544          ELSEIF ((UDELTA(I).LT.U_MIN(I)).AND.(U_MIN(I).LT.0.)) THEN
1545              SCALE = U_MIN(I)/UDELTA(I)
1546          ENDIF
1547      ENDIF
1548  ENDDO
1549 C
1550      SCALE = XSCALE*SCALE
1551 C
1552      DO I=1,M
1553          UMINNORM(I) = UKDA(I) + SCALE*(UP(I)-UKDA(I))
1554      ENDDO
1555 C
1556      RETURN
1557  END
1558
1559 C-----
```